CRWR Online Report 97-8

HEC-PREPRO:
A GIS Preprocessor for Lumped Parameter Hydrologic Modeling Programs

by

Ferdinand Leberect Hellweger, B.S.

Graduate Research Assistant

and

David R. Maidment, PhD

Principal Investigators

August 1997

CENTER FOR RESEARCH IN WATER RESOURCES

Bureau of Engineering Research • The University of Texas at Austin
J.J. Pickle Research Campus • Austin, TX 78712-4497

This document is available online via World Wide Web at
http://www.ce.utexas.edu/centers/crwr/reports/online.html

HEC-PREPRO:

A GIS Preprocessor for Lumped Parameter Hydrologic Modeling Programs

by

Ferdinand Leberecht Hellweger, B.S.

Thesis

Presented to the Faculty of the Graduate School

of The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

August 1997

HEC-PREPRO:

A GIS Preprocessor for Lumped Parameter Hydrologic Modeling Programs

APPROVED BY

SUPERVISING COMMITTEE:

_____

_____

ACKNOWLEDGMENTS

This Research was sponsored by the Hydrologic Engineering Center of the United States Army Corps of Engineers, Davis, California and supervised by David R. Maidment. Their support is gratefully appreciated.

July 10, 1997

HEC-PREPRO:

A GIS Preprocessor for Lumped Parameter Hydrologic Modeling Programs

by

Ferdinand Leberecht Hellweger, M.S.E.

The University of Texas at Austin, 1997

SUPERVISOR: David R. Maidment

This thesis documents a procedure called HEC-PREPRO, which was developed by the author. The purpose of the procedure is to automate the translation of data from a geographic information system (GIS) data structure to a hydrologic data structure used by lumped parameter hydrologic modeling programs. HEC-PREPRO is capable of identifying seven hydrologic elements (subbasins, reaches, sources, sinks, junctions, diversions and reservoirs) as well as connectivity among them. The procedure was implemented in ARC/INFO's Arc Macro Language (AML) and ArcView's Avenue programming language. In both implementations the data are written to an ASCII file which can be read by the hydrologic modeling program. Four system applications are documented.

TABLE OF CONTENTS.

LIST OF TABLES.

LIST OF FIGURES.

1. INTRODUCTION.

1.1. Background.

As part of the hydrologic cycle water precipitates out of the atmosphere and falls onto the land surface. Some of that water re-enters the atmosphere through evaporation or transpiration and a portion infiltrates into the soil. The remainder flows to the oceans as overland flow or runoff.

There are many instances when we are concerned with this runoff. When planning a flood control project, for example, we want to predict the flow in a river for a given precipitation. To predict the precipitation-runoff relationship a hydrologic model is constructed.

A hydrologic model is a tool for predicting the precipitation-runoff part of the hydrologic cycle. The model consists of data and software. The data is a description of the watershed developed from existing data or calibrated against observed flow. The software can be customized and specific to the project or company, or an off-the-shelf program developed and maintained by another agency. The model can be used to calculate flows for a certain precipitation.

A hydrologic model can classified based on its spatial resolution into two types: distributed or lumped parameter. In general, the distributed parameter model has a finer spatial resolution and thus more spatial computational units. Rather than making the distinction based on number or size of computational units a distinction can be made as follows. A lumped parameter hydrologic model can distinguished from a distributed parameter model by having a spatial resolution that allows for the manipulation of the shape, parameters and computational methods of the individual hydrologic computational units. If the computational units are too numerous to consider each single element as unique the model can be considered a distributed model.

For the lumped parameter hydrologic modeling approach, parts of the watershed are combined or lumped into individual hydrologic elements functioning as black box elements. Those black box elements are ordered into a model schematic of the watershed. This task is a data conversion process. Watershed data are converted from a geographic to a hydrologic data structure.

This thesis documents a procedure developed to automate the conversion of watershed data from a geographic to a hydrologic data structure. It is

designed to be implemented in a geographic information system (GIS) and has been implemented in the ARC/INFO and ArcView environments. The procedure has been applied to several watersheds. Four of those applications are presented. They are the Niger River watershed, the State of Texas watersheds, the 1993 Midwest Flood watershed and the City of Austin, Texas watersheds.

1.2. Constructing a Lumped Parameter Hydrologic Model.

The construction of a lumped parameter hydrologic model consists of two steps. First, a description of the watershed with a geographic data structure is constructed. This step consists of compilation and development of data needed to support the hydrologic calculations. Second, parts of the watershed are combined into hydrologic elements. Depending on the hydrologic element type and modeling method(s) to be used each hydrologic element is assigned corresponding hydrologic properties. This step also includes establishing connectivity of the hydrologic elements. In the hydrologic calculations the hydrologic elements are treated as black boxes that receive input and produce output which gets passed on as input to the next

downstream hydrologic element. A more detailed description of the two steps is presented below.

1.2.1. Step 1. Construct a Description of Watershed with Geographic Data Structure.

The building of a lumped parameter hydrologic model begins with the construction of a description of the watershed with a geographic data structure. For several data a choice of existing data sets might exist or other data sets might be available to develop the data from. The subbasins, for example, can be defined several different ways:

- Use a delineation produced by another party, like the Hydrologic Unit Codes (HUCs).
- Delineate subbasins from stream junctions using elevation data.
- Delineate subbasins from stream gaging stations using elevation data.
- Delineate subbasins based on the variation of surface properties, like slope or land use using elevation data.

- Delineate subbasins based on a surface area threshold using elevation data.
- Use any combination of the above.

If these data are presented on a map or in a GIS the watershed is said to be described with a geographic data structure. Figure 1.1 illustrates such data description.

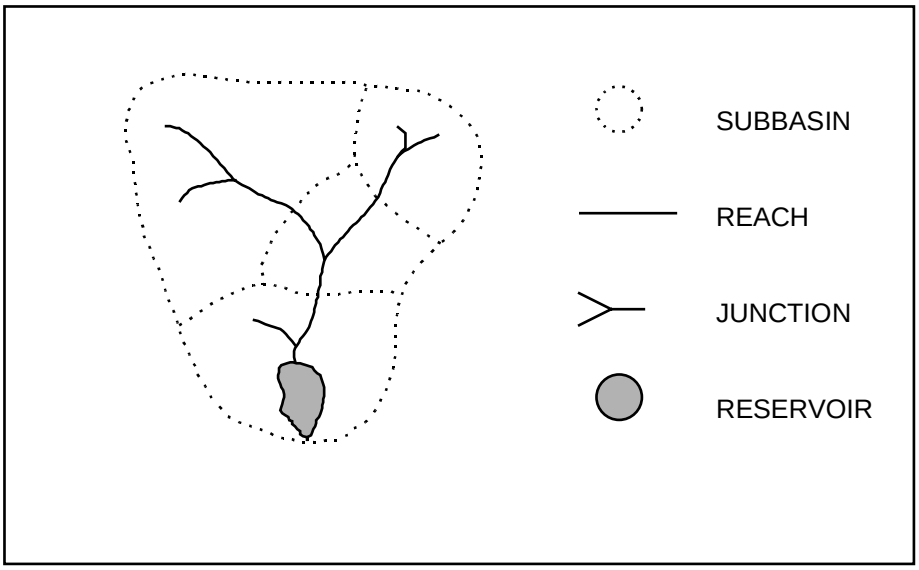

Figure 1.1. Sample Watershed Described with Geographic Data Structure.

Note that there are several data, like stream slope or observed hydrographs, which are not easily presented on a map. Those data can usually be connected to their corresponding geographic features as attributes. If that is done those data are also said to be described with a geographic data structure as well.

1.2.2. Step 2. Convert Watershed Data to Hydrologic Data Structure.

The second step in the procedure is to convert the watershed data to a form which is used as input to the modeling software. A lumped parameter hydrologic model operates on watershed data described with a hydrologic data structure. The hydrologic data structure consists of hydrologic elements including properties needed to support hydrologic modeling methods and the connectivity among those hydrologic elements.

First the hydrologic element type or class of each set of geographic features has to be established. This dictates the modeling methods available to convert the inflow to the outflow. A subbasin, for example, has a different set of modeling methods than a reach. Then the properties or attributes needed to support the modeling method have to be connected to the

hydrologic element. Depending on the modeling method a different set of properties is needed. Routing a flow through a river might, for example, require the Manning's n value whereas to determine the flow of water from a subbasin the surface area is an important property. Once the hydrologic elements have been defined connectivity of the hydrologic elements has to be established. This dictates the sequence of calculations in the hydrologic model.

Once the hydrologic elements have been defined and their connectivity is established the watershed data is said to be described with a hydrologic data structure. This data structure can be presented in a stick diagram form as shown in Figure 1.2 or in the form of an ASCII input file for a hydrologic model.

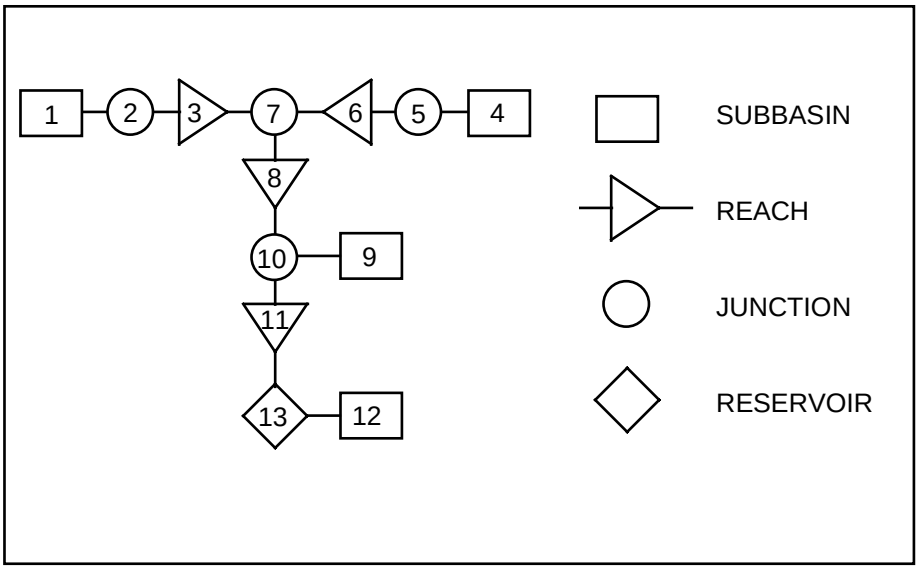

Figure 1.2. Sample Watershed Described with Hydrologic Data Structure.

1.3. Geographic Information Systems (GIS) and Hydrologic Modeling.

A geographic information system (GIS) is a collection of computer hardware and software, data and skilled personnel for managing and analyzing geographic data. The Environmental Systems Research Institute, Inc. (ESRI), a GIS software development company, offers the following definition (1997):

> A geographic information system (GIS) is a computer-based tool for mapping and analyzing things that exist and events that happen on Earth. GIS technology integrates common database operations such as query and statistical analysis with the unique visualization and geographic analysis benefits offered by maps. These abilities distinguish GIS from other information systems and make it valuable to a wide range of public and private enterprises for explaining events, predicting outcomes, and planning strategies.

In recent years the description of the environment in digital form has greatly improved in quality and quantity. Further, the tools that are available for analyzing these data have improved as well. This has led to an increase in the use of GIS for hydrologic modeling. For more detailed overviews of the use of GIS for hydrologic modeling Maidment (1996), DeVantier and Feldman (1993) and Moore (1993) should be consulted.

1.4. The Hydrologic Modeling System (HEC-HMS).

The Hydrologic Modeling System (HEC-HMS) is a computer software system for precipitation-runoff modeling. It is being developed by the Hydrologic Engineering Center (HEC) of the US Army Corps of Engineers in Davis, California. The system is currently in beta testing and available for Microsoft Windows and X-Window platforms. The system has a graphical user interface, which allows the user to draw the hydrologic elements on the screen and access their attributes by identifying the element graphically. It is intended for HEC-HMS to replace HEC-1, a popular hydrologic modeling software written in FORTRAN.

One of the requirements of HEC-HMS is an “interface to Geographic Information Systems” (Charley et al., 1995). For more information HEC (1996) and Peters (1995) should be consulted.

1.5. Motivation.

In recent years the usage of geographic information systems (GIS) for hydrologic modeling has increased. It is clear that GIS is going to play a significant role in hydrologic modeling. That is the reason why one of the requirements for HEC-HMS is to support GIS. The motivation of this study was to make an interface from HEC-HMS to GIS.

1.6. Objective.

The objective of this study was to develop a procedure for converting watershed data from a geographic to a hydrologic data structure. The procedure should be independent of the specific GIS software. The final

output of the procedure was to be watershed data presented in the form of an ASCII input file to HEC-HMS. The procedure should be implemented in the ARC/INFO and ArcView GIS environments.

1.7. Scope.

The scope of this study was limited. The main focus was on developing and implementing into a GIS environment, a procedure for converting watershed data from a geographic to a functional data structure. Although the development of a few hydrologic parameters is included in one of the implementations, the procedure is not designed around hydrologic data development. The hydrologic data development should be done prior to the data conversion using a system like that developed by Smith and Maidment (1995) or another system.

2. PREVIOUS RESEARCH.

2.1. Introduction.

This chapter summarizes literature reviewed as part of the study. Published literature on the subject of using GIS to construct input for lumped parameter hydrologic modeling programs was reviewed.

The connection of GIS and hydrologic modeling has been accomplished in several different ways. Distributed parameter hydrologic models have been constructed on raster GIS data and solved using finite difference or finite element methods (Stuebe and Johnson 1990; Saghafian 1996). For lumped parameter hydrologic models, the GIS usually plays the role of a preprocessor which translates data from the data structure of the GIS to the data structure of the hydrologic model (Fisher 1989; Shea et al. 1993).

2.2. Bhaskar et al. (1992).

Bhaskar et al. used ARC/INFO to estimate hydrologic parameters for the watershed hydrology simulation (WAHS) model. ARC/INFO commands were used to directly determine most parameters needed by WAHS.

Examples of those parameters are individual stream length, watershed area and watershed perimeter. Computation of other parameters required significant user interaction. The distance from individual first order streams to the watershed outlet, for example, had to be computed by manually identifying the individual lines comprising the flow path. Then the length of the individual lines were summed to yield the total length. Other parameters, like the beginning and ending elevation of streams, were not computed inside the GIS, but rather entered manually.

2.3. Engineering Computer Graphics Laboratory (1997).

Jim Nelson and Norm Jones of the Engineering Computer Graphics Laboratory (ECGL) at Brigham Young University (BYU) have developed and support a system called Watershed Modeling System (WMS) (1997). WMS is a stand-alone GIS system, not requiring any of the commercially available GIS systems, like ESRI's ARC/INFO. It has a graphical user interface (GUI) and is available in UNIX and PC versions.

The system writes and maintains input files for lumped parameter hydrologic modeling programs, like HEC-1 and SCS TR-20, in the

background as the user develops the hydrologic data in the GIS. WMS supports vector, raster and Triangulated Irregular Network (TIN) data structures. The system automatically determines the connectivity of the basic hydrologic elements, like subbasins and reaches.

2.4. Innovative System Developers, Inc. (1995).

Innovative System Developers, Inc. developed an ARC/INFO pre- and post-processor for the TR-20 hydrologic modeling program. The system constructs a complete TR-20 input file. However, some user interaction is required. For example, the user has to provide the system with the possible locations of upstream ends of the longest flow path for each subbasin. The system then computes the time of concentration for each of these points and uses the one with the largest time.

2.5. Shea et al. (1993).

Shea et al. constructed a system linking the Aeronca Electronics Geographic Information System (AEGIS) and AutoCAD to HEC-1 and HEC-

2. Automated procedures extracted data like the length and upstream and downstream end elevations of streams. Other data like subbasin connectivity were entered manually.

2.6. Smith and Maidment (1995).

Smith and Maidment have developed an ARC/INFO based system called Hydrologic Data Development System (HDDS). The system is written in ARC/INFO's Arc Macro Language (AML). The location of subbasin outlets and boundaries are determined automatically and parameters like average subbasin slope, time of concentration and SCS curve number are established. The data can be written to an input file readable by the Texas Department of Transportation Hydraulic Program (THYSYS), a hydrologic modeling software package.

2.7. Warwick and Haness (1994).

Warwick and Haness used ARC/INFO to determine hydrologic parameters for the HEC-1 hydrologic model. ARC/INFO commands were used to determine most of the required parameters directly (e.g. subbasin

area). A separate line coverage defining the runoff routes was created manually.

2.8. Summary.

A significant amount of work has been done on connecting GIS to lumped parameter hydrologic modeling programs. Several systems have been developed that extract data needed to support hydrologic modeling and create an input file readable by the hydrologic modeling program. Most of those data have been attribute values or properties of hydrologic elements. The definition of the hydrologic elements and their connectivity among them was usually very simple (i.e. no diversions or reservoirs) and had to be manipulated manually. It is the focus of this study to automatically handle the definition and connectivity of hydrologic elements.

3. METHODOLOGY.

3.1. Introduction.

This chapter describes the general methodology for converting watershed data from a geographic to a hydrologic data structure. First the two data structures are defined followed by a step-by-step outline of the conversion procedure. The methodology is presented at a level of detail which is detailed enough for implementation into a GIS system, but not too detailed to limit the implementation to a specific GIS system. Detailed input data requirements and computational methods are presented in the chapters describing the system implementations.

3.2. Geographic Data Structure Definition.

This section defines the geographic data structure in general terms. A more detailed definition is required for each specific system implementation. The description of watershed data with a geographic data structure is not unique. In other words a watershed can be described with a geographic data structure in several different ways. Consider for example the description of streams. Several different ways of defining streams are presented below.

- Single Line. Streams are described by a single polyline along the centerline of the stream. A polyline is a set of straight lines defined by a list of points. The order of the points in the list defines the direction of the polyline. This format is used in the EPA's River Reach File 1 (RF1) for inland streams. It is also the format upon which HEC-PREPRO is based.

- Double Line. Streams are described by a double polyline along the shorelines of the stream. This format is used by the USGS' Digital Line Graph (DLG) file for larger streams.

- Multiple Lines. Multiple lines along the shoreline representing different water surface elevations. This format is used by the USGS' DLG file for areas where the shoreline is significantly variable depending on the time of year.

- Grid Cells. A sequence of valued grid cells following the general path of the stream. A data set of this type can be achieved from an elevation grid of different cell size. A drainage area threshold has

to be defined to determine the location of the first cell in this sequence.

If the data are described in a way which is not desirable they can usually be converted using the GIS capabilities. For example: To convert the grid cell representation of a stream line to a single line representation the ARC/INFO GIS system offers the STREAMLINE function.

The geographic description of the watershed has to be free of hydrologic errors. A hydrologic error is different than a geographic error as described below.

- Geographic Error. The location or shape of a geographic feature is misrepresented. A stream that is shifted from its actual location is an example of a geographic error.

- Hydrologic Error. The geographic features in the watershed are presented in a way which is wrong from a hydrologic point of view. A stream that zigzags across a subbasin boundary is an example of a hydrologic error.

More discussion on errors in the input data are presented in the system implementation and application chapters. In the geographic data structure used by HEC-PREPRO subbasins, streams, reservoirs and elevations are described as follows.

- Subbasins. Described with a polygon layer. The bounding watershed is completely subdivided into non-overlapping subbasins.

- Streams. Described with a polyline layer. A single polyline runs along the centerline of the stream in the downstream direction.

- Reservoirs. Described with the streams in the stream layer. A polygon in the stream layer is considered a reservoir.

- Elevations. Described with an elevation grid.

Thus, three data layers are needed for HEC-PREPRO: a line coverage of streams and reservoirs, a polygon coverage of subbasins, and a elevation grid to describe the land surface terrain.

3.3. Hydrologic Data Structure Definition.

This section defines the hydrologic data structure used by the HEC-PREPRO procedure. As with the geographic data structure the definition of a hydrologic data structure is not unique. Different lumped parameter hydrologic modeling software programs can use a different data structure. Consider for example subbasins and streams. In HEC-HMS they constitute two different hydrologic elements. In the TR-55 model the subbasins and their downstream reach(es) are combined into one element forming a balloon-like hydrologic element.

The data structure used in this study is identical to the one used by the HEC-HMS model and is defined by HEC (1996) as follows:

> Subbasin. A subbasin is conceptually an element that produces a discharge hydrograph at its outlet. Its properties include area and percent imperviousness. The discharge hydrograph is based on subtracting "losses" from input precipitation, transforming the

resulting precipitation excess to direct runoff at the outlet, and adding baseflow. If the modClark transform is used (with gridded rainfall), it is also necessary to access, by means of cell-parameter file, characteristics of subbasin grid cells.

River Reach. A river reach is conceptually a linear element for which there is a "known" discharge hydrograph at its upstream end, and which produces a discharge hydrograph at its downstream end. Data requirements vary from a single parameter for the simplest routing method to specification of a representative cross section and channel properties for more complex methods.

Junction. A junction is a location where two or more inflow hydrographs are added together to produce an outflow hydrograph.

Reservoir. A reservoir is similar to a routing reach in that there is a "known" discharge hydrograph that depicts inflow to the reservoir, and the reservoir element produces an outflow hydrograph. In the current version of HEC-HMS, capability only exists for routing through an uncontrolled reservoir, for which there is a monotonically increasing relationship between reservoir storage and outflow.

Diversion. A diversion is an element for which a portion of the inflow to the element is diverted out, and the remainder passes through. The diversion is based on a user-specified relationship between inflow and diverted flow. The diverted flow can be brought back into the basin network at a hydrologic element that is conceptually downstream from the point of diversion.

Source. A source is an element with which a discharge hydrograph is imported into the basin network. The element might be used to import an observed hydrograph, or a hydrograph generated in a prior simulation.

Sink. A sink is an element for which there is an inflow but no outflow.

3.4. Procedure.

The step-by-step procedure for converting watershed data from a geographic to a hydrologic data structure is presented in this section. The procedure is general and is independent of the GIS system used.

Details of the procedure vary from implementation to implementation as discussed in the corresponding chapters. Consider for example the definition of the location of a junction element. In the ARC/INFO implementation of the procedure, a junction element has a topological feature associated with it called a node. Its relationship to the surrounding stream lines is defined by means of the stream lines having the node number as an attribute. In the ArcView implementation a junction element has a point feature associated with it. Its connection to the stream network is established by checking the distance to the endpoints of stream lines. If the distance to the endpoint of a line is within a certain tolerance of the end or starting point of the line, they are connected. These differences occur because the data structure of the shape files used by ArcView is simpler than that of the line coverage used by ARC/INFO.

The procedure is presented using a sample data set. The data is a modified version of the Tenkiller Reservoir watershed located in North East Oklahoma and North West Arkansas. Some artificial features were added to the natural features of this watershed so that all seven hydrologic elements would be present in this example.

3.4.1. Prepare Input Data.

The procedure begins with a stream and a subbasin layer shown as solid and dotted lines, respectively, in Figure 3.1. The watershed is divided into four subbasins. One stream in the upper right subbasin was extended beyond the watershed boundary to simulate a source element. A lake, including an extra stream entering and leaving the lake, was added to the upper left subbasin. Finally, a diversion was added in the lower subbasin.

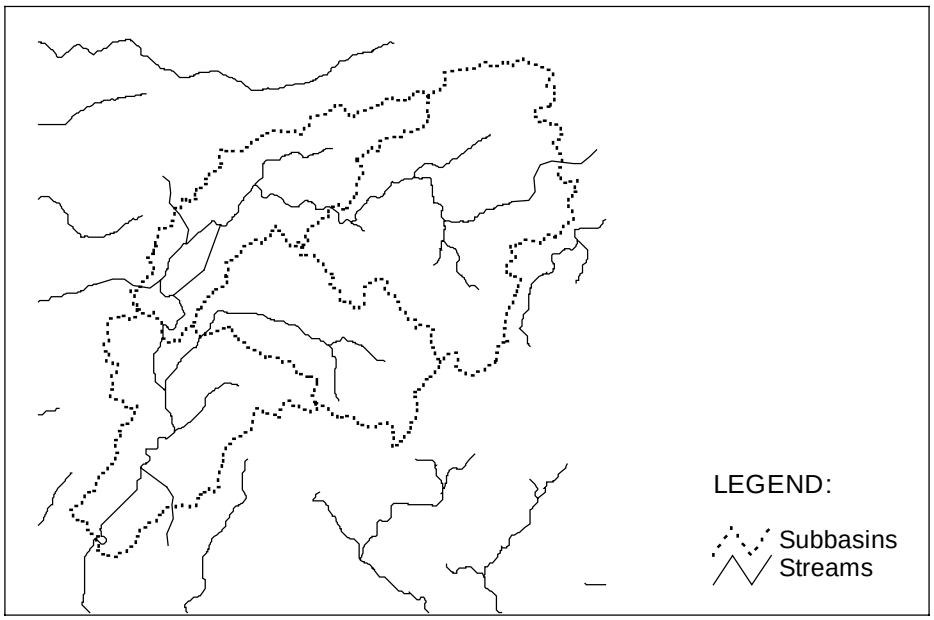

Figure 3.1. Input Stream and Subbasin Layers.

3.4.2. Intersect Stream and Subbasin Layers.

The stream layer is intersected with the subbasin layer to produce a new layer containing only those streams within the watershed as shown in Figure 3.2. This process removes streams outside the watershed and identifies the intersection of streams and subbasin boundaries. The intersections represent the locations of sources, subbasin outlets or sinks. The intersections are marked with solid triangles in Figure 3.2.

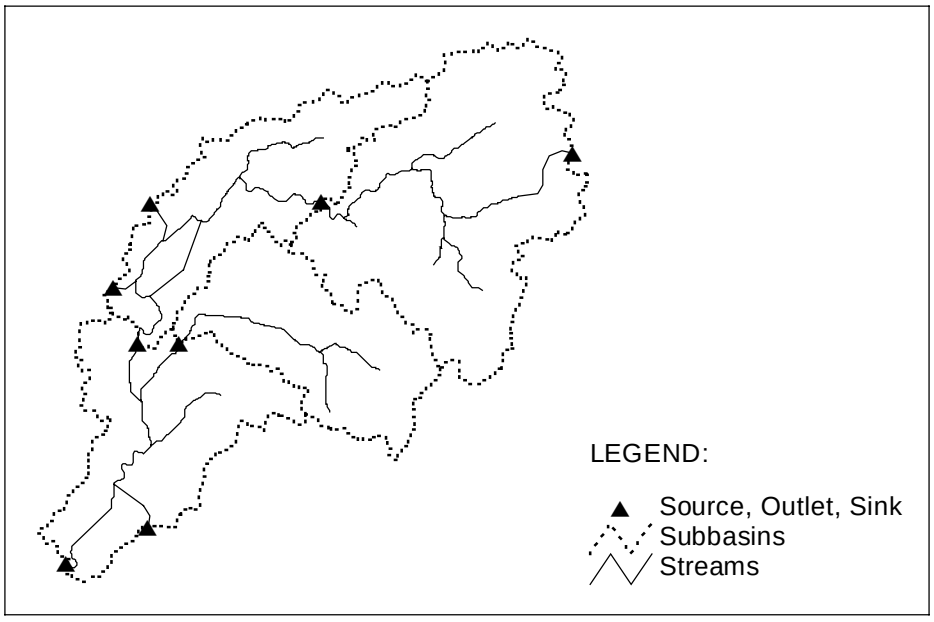

Figure 3.2. Result of Stream and Subbasin Layer Intersection.

3.4.3. Identify Channel System and Lakes.

Stream lines are classified into three types: (1) Lines that are part of the channel system carrying water from upstream features, (2) lines that are tributaries to this channel system, and (3) lines that are part of a reservoir defined by double-line connections between two channel nodes.

The lines forming the channel system are those downstream of hydrologic elements. There is always a source or a subbasin outlet at the most upstream end of a channel system, which means that the channel system can also be defined by being downstream of sources and subbasin outlets. The lines forming the channel system are identified by tracing downstream of the source, sink and subbasin outlet locations identified in the previous step.

A double-line connection between two locations on the stream defines an area or polygon which is commonly used to represent reservoirs whereas streams are usually represented with lines. Reservoir lines are identified by being part of enclosed polygons in the stream coverage.

The channel stream lines (heavy solid lines), the non-channel stream lines (light solid lines) and the reservoir outline (heavy dashed lines) are shown in Figure 3.3.

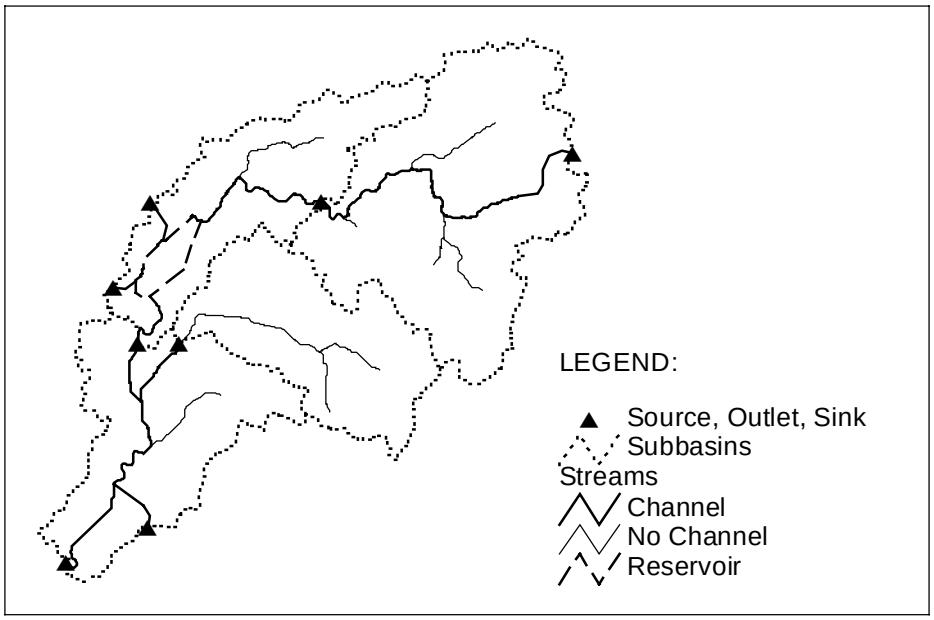

Figure 3.3. Result of Stream Line Classification.

3.4.4. Identify Channel Elements.

All the elements on the channel can be identified based on their relation to the connecting lines. For example: a node having multiple

upstream lines is defined as a junction; a node having multiple downstream lines is defined as a diversion. Other functions a node can take on for a stream is being the most upstream or downstream part or being an interior (pseudo) node. The incorporation of lakes and reservoirs adds another five possibilities to set of functions a node can take on. A node located on a reservoir can be at the most upstream or downstream part of the reservoir, constitute an interior node or a junction or diversion out of the reservoir. There are a total of 10 functions a node can take on, as shown in Figure 3.4. This node-line topology provides a description of the hydrologic behavior of each node with respect to its immediate surroundings or on a micro-scale.

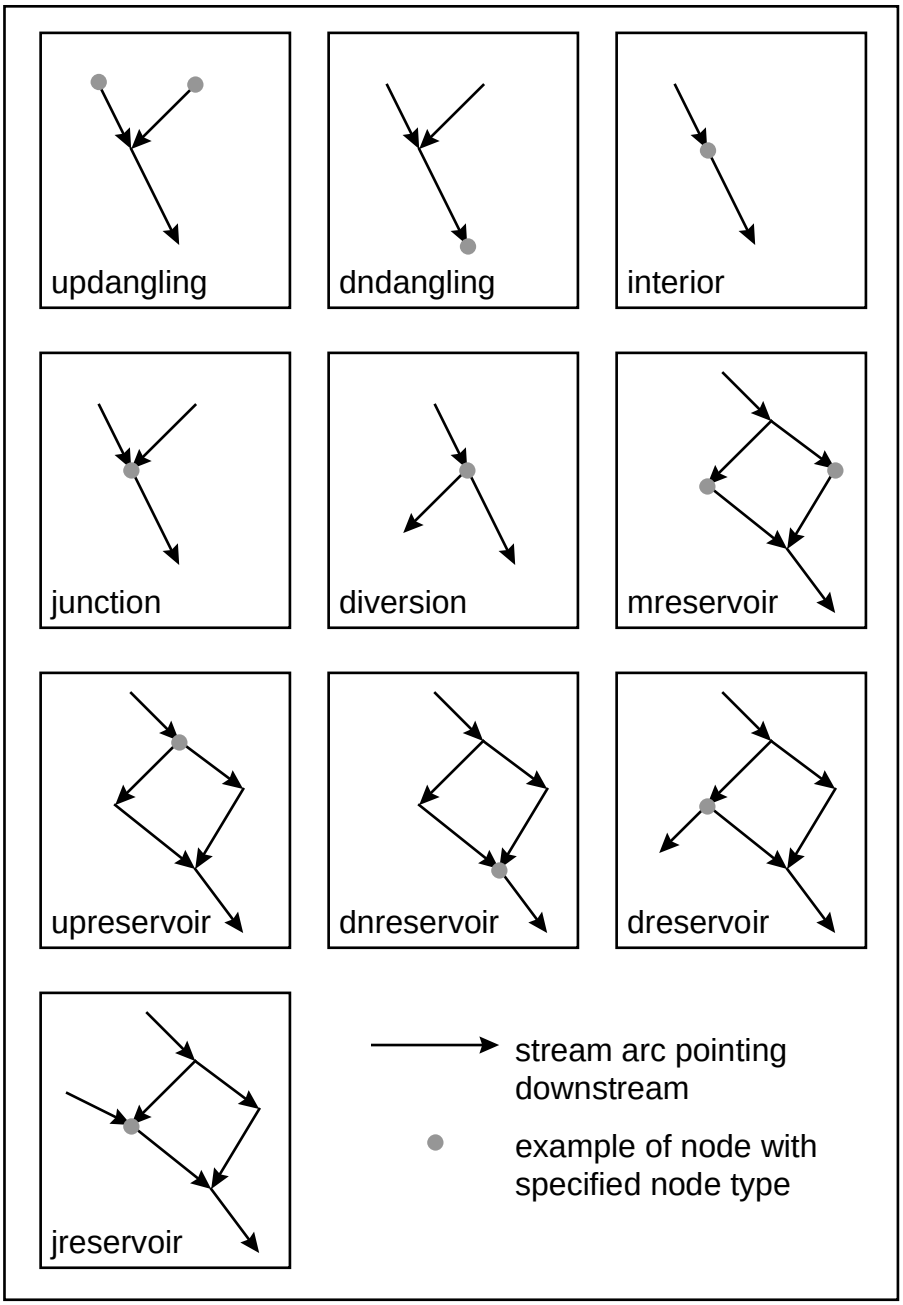

Figure 3.4. Node Type Illustration.

Note that this node-line topology can be calculated taking into account all the stream lines or only the lines belonging to the channel system. There are therefore two ‘types’ a node can take on. Consider, for example, the subbasin in the upper right corner of Figure 3.3. When taking all the lines into account there are four ‘junctions’ whereas when taking only lines from the channel system into account there are only ‘interior’ nodes. Also note that this assumes that streams are defined by single and reservoirs by double lines. See Table 3.1.

Node Type	Description
unknown	unknown
none	none
updangling	a node at the upstream end of a dangling line.
dndangling	a node at the downstream end of a dangling line.
interior	a node connecting two lines pointing in the same direction (a pseudo node).
junction	a node which has multiple lines pointing towards it and one line pointing away.
diversion	a node which has multiple lines pointing away from it and one line pointing towards it.
mreservoir	a node connecting two lines along a reservoir pointing in the same direction (a pseudo node).
mpreservoir	a node at the most upstream end of a reservoir
dnreservoir	a node at the downstream end of a reservoir.
dreservoir	a node at a reservoir diversion.
jreservoir	a node at a reservoir junction.

Table 3.1. Node Type Description.

With this node-line topology the information can be further reduced to identify the functioning of each node in the overall flow network using. Diversions (hollow circles), junctions (solid circles), reservoirs (hollow triangles), sinks (upside-down solid triangles), sources (upside-down hollow triangles), and subbasin outlets (solid triangles) are shown in Figure 3.5.

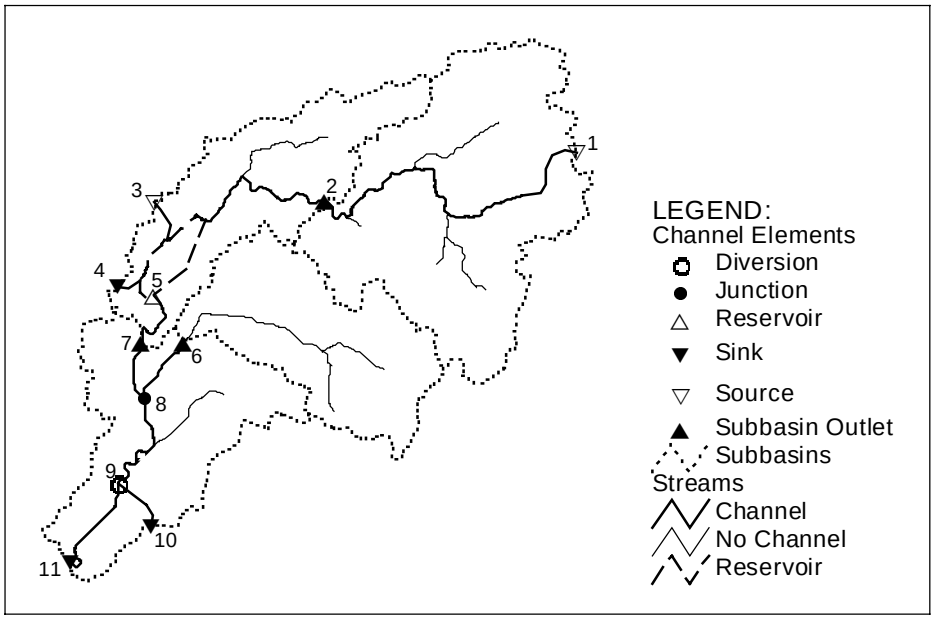

Figure 3.5. Result of Channel Element Identification.

3.4.5. Calculate Connectivity.

Subbasin elements are defined by the centroid of the polygons in the subbasin coverage. To establish connectivity among subbasin elements and subbasin outlets, the stream line upstream of the subbasin outlet is examined to identify in which polygon they are located. Subbasin elements are assigned a unique ID.

Connectivity among channel elements is established by moving along the channel streams from element to element. During this process multiple stream lines are combined into single reaches, that is, the lines that make up a particular reach are all assigned the same ID number. See Figure 3.6.

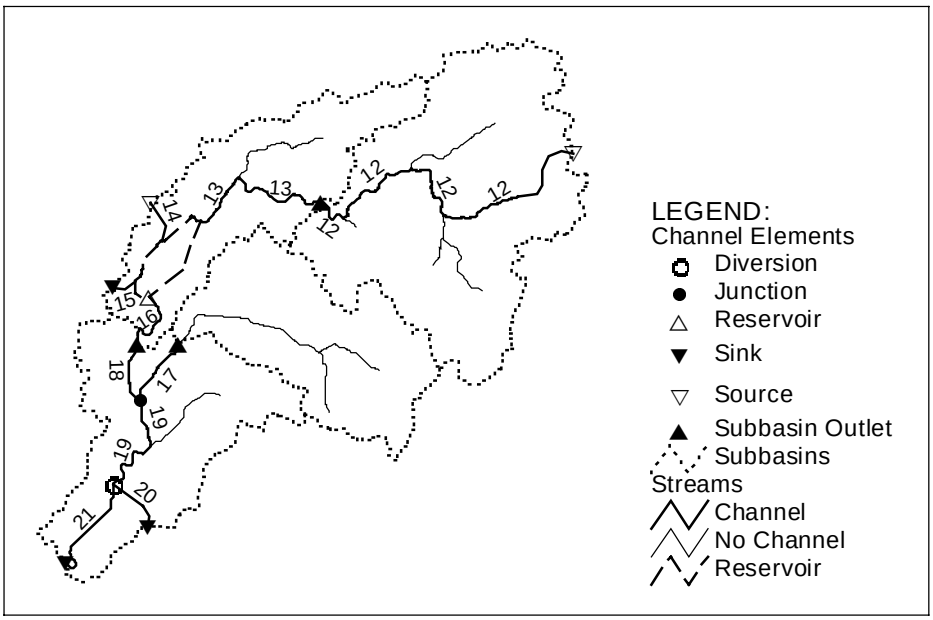

Figure 3.6. Result of Connectivity Calculation.

3.4.6. Create Symbolic Layer.

With all the channel elements established and connectivity identified, a symbolic layer is generated showing the schematic model of the flow network, as shown in Figure 3.7. The lines in the layer represent connections among hydrologic elements. A connection can be via a reach (heavy lines) or a link (light lines) from a subbasin to its outlet. The nodes in the coverage represent connections between channel elements. Channel elements can be diversions (hollow circles), junctions (solid circles), reservoirs (hollow triangles), sinks

(upside-down solid triangles), sources (upside-down hollow triangles), and subbasins (rectangles). Note that elements previously classified as subbasin outlets are now combined with junctions, because in the hydrologic data structure they both serve as flow summing points.

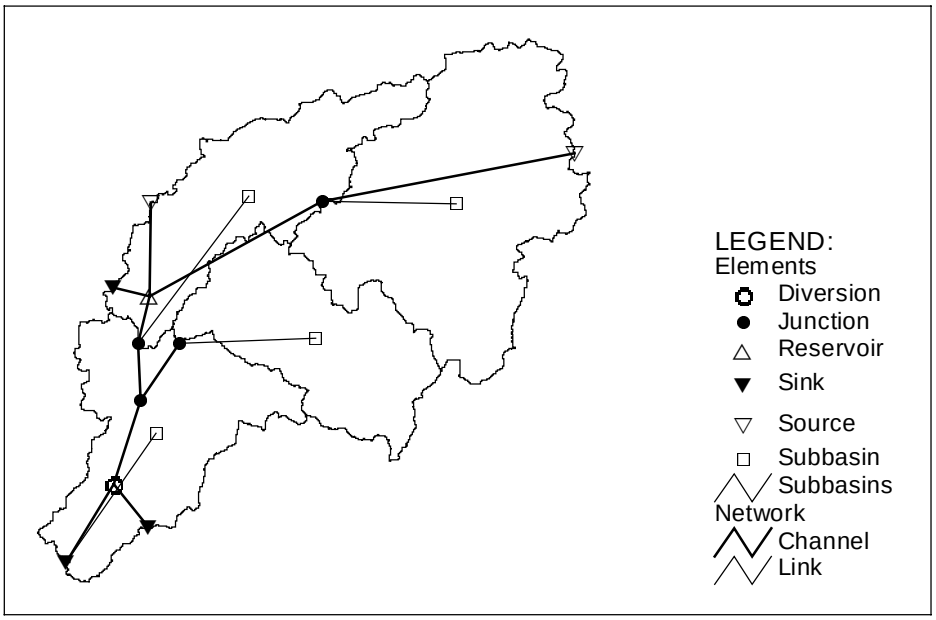

Figure 3.7. Symbolic Layer.

3.4.7. Create HEC-HMS Basin File.

The identification numbers of the hydrologic elements are written to the attribute tables of the input layers to allow for the relation back to the

geographic data. The main purpose of the symbolic layer is for the user to check the results of the system. The schematic data model is written to a basin file readable by HEC-HMS. Figure 3.8 shows part of an HEC-HMS basin file.

```
Basin: HECPREPRO generated HMS basin file
     Description: HECPREPRO generated HMS basin file
     Last Modified Date: 10 June 1996
     Last Modified Time: 12:58
     Unit System: Unknown
End:

Subbasin:  23
     Canvas X:  115488.609
     Canvas Y:  1455012.500
     Downstream:  7
     Area:  845179044
End:

...

Source:  1
     Canvas X:  169282.81250
     Canvas Y:  1462399.39734
     Downstream:  12
End:

...
```

Figure 3.8. Parts of an HEC-HMS Basin File (continued on next page).

```
...

Reach:  12
     Downstream:  2
End:

...

Junction:  2
     Canvas X:  127482.82031
     Canvas Y:  1454141.12500
     Downstream:  13
End:

...

Reservoir:  5
     Canvas X:  99024.17188
     Canvas Y:  1438141.12500
     Downstream:  16 15
End:

...

Diversion:  9
     Canvas X:  93124.17188
     Canvas Y:  1406660.84829
     Downstream:  20 21
End:

...

Sink:  11
     Canvas X:  85182.82031
     Canvas Y:  1393441.12500
End:
```

Figure 3.8. Parts of an HEC-HMS Basin File (continued from previous page).

4. IMPLEMENTATION OF HEC-PREPRO IN ARC/INFO.

4.1. Introduction.

The methodology was implemented in the ARC/INFO environment. ARC/INFO is UNIX based GIS software developed by the Environmental Systems Research Institute (ESRI, 1996a). ARC/INFO provides for a programming environment via a macro language called Arc Macro Language (AML). This chapter describes the implementation of the methodology in ARC/INFO AML.

First the input data requirements are defined followed by a detailed step-by-step procedure and a description of the output data. Lastly, a section on using the system serves as a user's guide.

4.2. Input Data Description.

This section describes the input data requirements in detail. As input a stream coverage, subbasin coverage and elevation grid are required. Those data sets have to be in the same projection and datum. The input data has to accurately describe the hydrologic properties of the area. Where errors occur

they are due to discrepancies among the stream and subbasin coverages. An example of an error in the input data is shown in Figure 4.1. There a stream is zigzagging across the subbasin boundary. Another example of an error is shown in Figure 4.2. There Grid routines were used to delineate the basin boundary. Subsequent conversion from grid to coverage created the small enclosed subbasin at the right basin boundary. That is an error commonly encountered on a small scale. Those errors have to be corrected before running the system. For this purpose the ARC/INFO ArcEdit module provides for a suitable editing environment. Also the Terrain module of GISHydro97 has programs which automatically detect and correct these and other hydrologic errors.

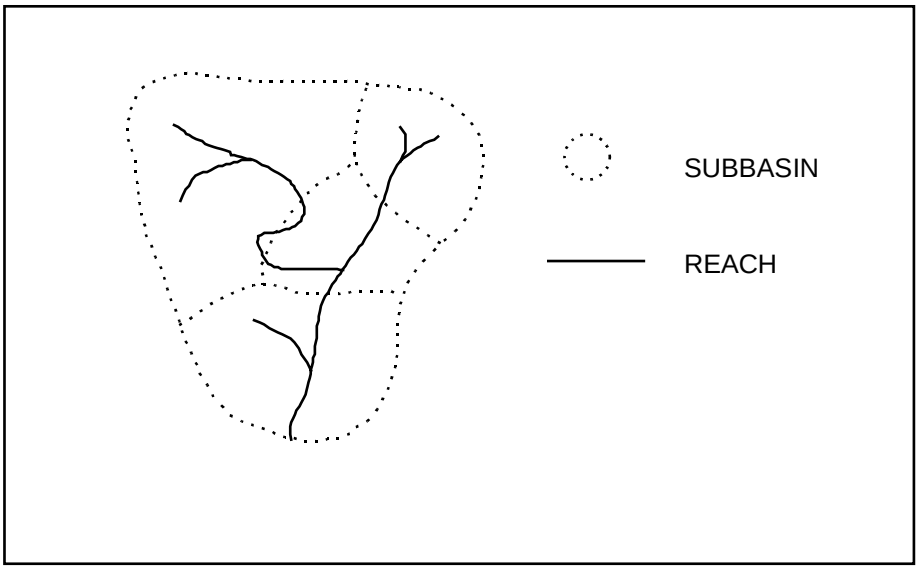

Figure 4.1. First Example of Error in Input Data (Stream Zigzagging Across Subbasin Boundary).

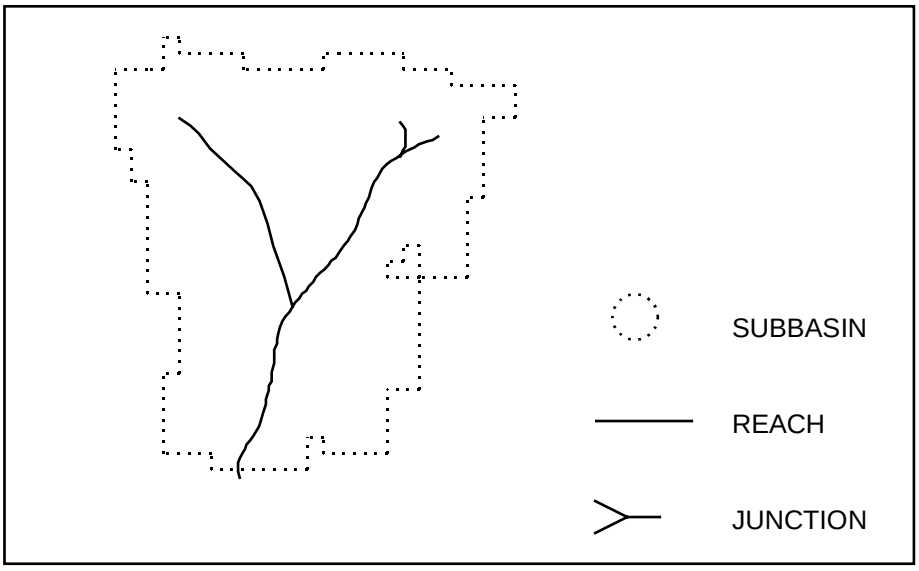

Figure 4.2. Second Example of Error in Input Data (Small Subarea of Watershed Separated From Remainder During Grid to Polygon Conversion).

The following sections describe specific input data requirements for each of the three data sets.

4.2.1. Stream Coverage.

- The stream coverage has to be an ARC/INFO line coverage.

- The coverage does not have to contain node, line or polygon topology. The HEC-PREPRO system will build the necessary topology.

- If the coverage has node topology the node attribute table cannot contain items named ‘hectype’, ‘hecid’, ‘hecupid’, ‘hecdnid’, ‘nodez’ or ‘upnode’, because these are created and used by HEC-PREPRO.

- If the coverage has line topology the line attribute table cannot contain items named ‘hectype’, ‘hecid’, ‘hecupid’, ‘hecdnid’, ‘heclength’ or ‘hecslope’, because these are created and used by HEC-PREPRO.

- The coverage has to be ‘clean’. To check if the coverage is clean use the arc DESCRIBE command. The DESCRIBE tells the user if the coverage was modified since the last time it was ‘cleaned’. If it was modified it needs to be ‘cleaned’. To ‘clean’ a coverage use the arc CLEAN command.

- See the discussion on the ‘node snapping option’ in Section 4.5.4.3 if the stream coverage has nodes near intersections of streams with the subbasin coverage. This is most often the result of delineating streams and/or subbasin boundaries with grid routines. Also check the fuzzy tolerance of the coverages if problems occur with node snapping. The arc DESCRIBE command can be used to display the fuzzy tolerance of a coverage.

- See the discussion on the ‘clipped option’ in Section 4.5.4.2 if the stream coverage has pseudo nodes at intersections with the subbasin coverage. This is most often the result of using the subbasin coverage to cut out an area from the stream coverage using the arc CLIP or INTERSECT command.

- All stream lines have to point in the downstream direction. This means that the from-node is upstream of the to-node. The direction of lines pointing in the upstream direction can be reversed with the ArcEdit FLIP command.

- Streams are represented by single lines. Double-line connections between two stream nodes will be interpreted as reservoirs, as described below.

- Reservoirs are represented by double-line connections between two stream nodes. There is only one main outlet to the reservoir which is defined by the node which has two lines that are part of the reservoir pointing towards it.

4.2.2. Subbasin Coverage.

- The subbasin coverage has to be an ARC/INFO polygon coverage.

- The coverage does not have to contain node, line or polygon topology. The system will build necessary topology.

- If the coverage has polygon topology the polygon attribute table cannot contain items named ‘hectype’, ‘hecid’, ‘hecdnid’, ‘cenz’,

'means', 'medians', 'b-coord', 'f-coord', 'fupz', 'flength' or 'flength2', because these are created and used by HEC-PREPRO.

- The coverage has to be 'clean'. To check if the coverage is clean, use the arc DESCRIBE command. The DESCRIBE tells the user if the coverage was modified since the last time it was 'cleaned'. If it was modified it needs to be 'cleaned'. To 'clean' a coverage use the arc CLEAN command.

- Each polygon in the coverage defines a subbasin.

- Each subbasin has to contain at least part of one feature of the stream coverage. In other words, each subbasin has to contain at least one stream.

4.2.3. Elevation Grid.

- The elevation grid has to be an ARC/INFO grid.

- The extent of the grid and the valued cells has to cover that of the stream and subbasin coverages.

4.3. Procedure.

This section outlines the methodology and procedure of the system. It is intended to document the system so that it can be modified, expanded or reproduced in another programming environment. In general the procedure is based on establishing the seven hydrologic elements and connectivity among them as discussed in Section 3.4. Attributes are collected and summarized where applicable, usually where the information is at hand. For this discussion a type convention as shown in Table 4.1 is used.

Type	Convention	Example
Commands	system COMMAND command	arc UNGENERATE command
Functions	system FUNCTION function	grid INT function
Directives	system DIRECTIVE directive	AML &SV directive
Data File	**datafile**	**streamcov.nat**
Table Item	**table**//item	**hydrocov.nat**//hecid
Variables:		
Single	*variablename*	*hecid*
Array	*arrayname*(n)	*hecx*(34)
Matrix	*matrixname*(n, m)	*hecupid*(2,56)

Table 4.1. Type Convention.

The processing is done in an AML file called hecprepro.aml. Several other utility programs are also available. All the AML programs are listed in Appendix A. The section numbering system used in this discussion is consistent with that in the program code. For clarity several parts of the system procedure are omitted in this discussion. They are:

- Echo statements to the command line.

- User observation level pauses.

- Display of the features in graphic windows.

- Maneuvering among different ARC/INFO modules (e.g. Grid, Tables).

- Error checking and tracking.

- Attribute collection option (it is assumed attributes are collected).

To keep the scope of the discussion manageable several repeated methods are also omitted in the discussion. They are:

- Assigning values to and determining values from variables, arrays and matrices. That is usually done with the AML &SV directive. The management of arrays and matrices is more complicated, since AML does not support them directly.

- Assigning values to and determining values from attribute tables. Several records can be assigned values with the arcplot CALCULATE command. Single records are mostly modified with general CURSOR commands.

- Managing selection sets. Selection sets are usually managed with general UNSELECT, ASELECT and NSELECT commands.

- Repeated processing in the form of loops. Most loops are closely related to attribute records and are therefore managed with general CURSOR commands and the AML &DO &UNTIL and &END directives.

4.3.1. Data Set Up.

The first step in the procedure is to prepare the input data for processing. This consists of such things as managing the command line input, making working copies of the coverages and adding working items to the attribute files.

4.3.1.1. Manage Command Line Input.

Nine variables are communicated to the program over the command line using the AML &ARGS directive. Only the first three variables specifying the input data sets are required. Other variables will be assigned default values if not specified. See Section 4.5.2 for details on the command line input.

4.3.1.2. Make Working Copies of Input Data Sets.

Working copies of the stream and subbasin coverages (**streamcov** and **subcov**) are made using the arc COPY command. This is done so that the original data is not modified. Previously existing versions of **streamcov** or **subcov** are deleted using the arc KILL ALL command. A working copy of the elevation grid is not made, because it is not modified.

4.3.1.3. Create Grid of Subbasins.

A subbasin grid (**subgrid**) is created from **subcov** using the arc POLYGRID command. The same cell size as in the elevation grid is used. This grid is used later in the program to find the mean and median surface slope of each subbasin. If necessary, previously existing versions of **subgrid** are deleted using the arc KILL ALL command.

4.3.1.4. Build Topology.

Node and line topology are created for **streamcov**, and node, line and polygon topology are created for **subcov** using the arc BUILD command. Topology is essential for defining connectivity among features. Polygon labels of **subcov** are moved to the centroid of the polygons using the arc CENTROIDLABELS command and the xy coordinates of the labels are added to the **subcov** polygon attribute table (**subcov.pat**) using the arc ADDXY command. XY coordinates of the subbasin centroids are needed, because the subbasin location is defined at that point.

4.3.1.5. Add Working Items to INFO Files.

Several working items are added to the attribute tables of **streamcov** and **subcov** using the arc ADDITEM command. Description of attribute items are presented in Table 4.2. Table 4.3 shows how the items are defined.

Attribute Table	Item	Description
streamcov.nat	hectype	feature hectype
	hecid	unique identification number
	hecupid	hecid of upstream element
	hecdnid	hecid of downstream element
	nodez	node surface elevation*
streamcov.aat	hectype	feature hectype
	hecid	unique identification number
	hecupid	hecid of upstream element
	hecdnid	hecid of downstream element
	heclength	length of reach*
	hecslope	slope of reach*
subcov.pat	hectype	feature hectype
	hecid	unique identification number
	hecdnid	hecid of upstream element
	cenz	hecid of downstream element
	means	mean surface slope*
	medians	median surface slope*
	b-coord	name of file holding coordinates of subbasin boundary
	f-coord	name of file holding longest flow path coordinates
	fupz	elevation of upstream end of longest flow path*
	flength	length of longest flow path*
	flength2	length along longest flow path from point nearest to subbasin centroid to subbasin outlet*

* Units are the same as those in the input data files.

Table 4.2. Working Item Description.

Attribute Table	Item	Width	Output Width	Type	Decimal Places
streamcov.nat	hectype	4	4	i	--
	hecid	4	4	i	--
	hecupid	4	4	i	--
	hecdnid	4	4	i	--
	nodez	8	8	f	2
	upnode	4	4	i	--
streamcov.aat	hectype	4	4	i	--
	hecid	4	4	i	--
	hecupid	4	4	i	--
	hecdnid	4	4	i	--
	heclength	8	8	f	2
	hecslope	8	8	f	6
subcov.pat	hectype	4	4	i	--
	hecid	4	4	i	--
	hecdnid	4	4	i	--
	cenz	8	8	f	2
	means	8	8	f	6
	medians	8	8	f	6
	b-coord	8	8	c	--
	f-coord	8	8	c	--
	fupz	8	8	f	2
	flength	8	8	f	2
	flength2	8	8	f	2

Table 4.3. Working Item Definition.

The hectype is used to classify the features according to the hectype system. The hectype system is a more thorough classification system than the seven element system. It is needed because the seven element system is not

detailed enough to establish connectivity among hydrologic elements. For example: In the seven element system, a reach confluence and a subbasin outlet are both classified as junctions. When establishing element connectivity it is important to know if the element is a subbasin outlet, because there will have to be a corresponding subbasin element. Table 4.4. lists the possible hectypes.

Hectype	Description
-1	unknown
0	not used
1	subbasin outlet
2	non-channel element*
3	junction
4	channel element*
5	subbasin
6	sink
7	source
8	diversion
9	not used
10	reservoir

* The channel system is defined by being downstream of hydrologic elements.

Table 4.4. Hectype Description.

The hecid is a unique identification number for each hydrologic element which is assigned by the system. The hecupid and hecdnid items are

used for storing the hecid of the upstream and downstream hydrologic elements, respectively. Hydrologic elements adjacent to one-another can be represented by two nodes (e.g. subbasin and junction) or a node and a line (e.g. diversion and reach).

4.3.1.6. Mark Elements as Unknown Hectype.

All the lines and nodes in **streamcov** are marked as unknown hectype by setting **streamcov.nat**//hectype and **streamcov.aat**//hectype of all records to -1.

4.3.2. Establish Source, Sink and Subbasin Outlet Elements.

The first elements that are established are sources, sinks and subbasin outlets. Those elements are located on streams where they intersect with the subbasin boundaries. The simplest way to establish these elements is to intersect the stream and subbasin coverages. Nodes representing sources,

sinks and subbasin outlet elements can then be differentiated from other nodes based on having been created by the intersection action.

In some cases the stream coverage already contains nodes representing sources, sinks and subbasin outlets near intersections with the subbasin boundaries. This is the case if the subbasins are delineated from the stream coverage using Grid routines. An option is provided to snap nodes from the stream coverage to nodes from the subbasin coverage. See the discussion in Section 4.5.4.3. In that case, nodes representing sources, sinks or subbasin outlets could have existed before the intersection. Nodes representing sources, sinks and subbasin outlet elements can then be differentiated from other nodes based on having been created by the intersection or based on having been snapped.

It is also possible that the subbasin coverage was used to clip the stream coverage from a larger stream coverage. See the discussion in Section 4.5.4.2. In that case the only way to determine source, sink and subbasin outlet elements is to look at each node of the stream coverage and determine if it is located at an intersection with the subbasin coverage.

4.3.2.1. Create Nodes.

Streamcov is intersected with **subcov** to create a new coverage (**hydrocov**) using the arc INTERSECT command. Figure 4.3. shows the arc syntax. The resulting **hydrocov** is the same as **streamcov** except that it has nodes at the intersections with the **subcov** lines. These nodes are sink, source or subbasin outlet elements.

```
intersect streamcov subcov hydrocov line # join
```

Figure 4.3. Arc Intersect Syntax.

The 'join' parameter in the syntax adds items from **subcov.pat** to **hydrocov.aat** if they do not already exist in **hydrocov.aat**. This means that there will be a subcov# item in **hydrocov.aat** specifying the polygon the line is located in. That information is used later to establish connectivity among subbasins and the channel system. If necessary, a previously existing **hydrocov** is deleted using the arc KILL ALL command.

These newly created nodes have ‘empty’ attribute values, which means they will have a hectype 0. That is the key to distinguishing them from the original nodes which were previously assigned a hectype -1 (Section 4.3.1.6). The source, sink and subbasin outlet elements can therefore distinguished based on having a **hydrocov.nat**//hectype 0 instead of -1

4.3.2.2. Snap Nodes.

If specified with the *snapit* variable, nodes in **hydrocov** within a certain distance (*snaptol*) of nodes in **subcov** are snapped to nodes in **subcov** using the arcedit SNAP command.

After the node snapping there could be nodes representing sources, sinks and subbasin outlets which existed prior to the intersection. Those nodes cannot be distinguished from the other nodes based on hectype, because they have a hectype -1. To identify sources, sinks and subbasin outlet elements the nodes that were snapped have to be distinguished from the other nodes. This is made possible by adding the ‘snapped’ item to **subcov.nat** and

hydrocov.nat, and setting **subcov.nat**//snapped to 1. It is then specified that the value of the snapped item will be transfered among snapped features with the arcedit SNAPITEMS command. This way all the nodes that were snapped will have **hydrocov.nat**//snapped equal to 1. The ARC/INFO syntax is shown in Figure 4.4.

```
arcedit
edit hydrocov
editfeature arc
snapcoverage subcov
snapfeatures node node
mape subcov
snapping closest snaptol
snapitems snapped
aselect all
snap
save
q
```

Figure 4.4. ARC/INFO Snapping Syntax.

4.3.2.3. Set Up Hydrocov for Further Processing.

Line, node and polygon topology for **hydrocov** is created using the arc CLEAN and BUILD commands. This is needed, because the coverage was modified by the intersection and possibly the node snapping. XY coordinates

of the nodes are added to **hydrocov.nat** using the arc ADDXY command. XY coordinates are needed, because they define the location of the elements.

4.3.2.4. Calculate Surface Elevation Attributes.

Subbasin Attributes: A new grid (**slopegrid**) storing the surface slope (in percent) is created from the elevation grid using the grid SLOPE function. Two new grids (**meansgrid** and **mediansgrid**) storing the mean and median surface slopes for each subbasin are created from **slopegrid** and **polygrid** using the grid ZONALMEAN and ZONALMEDIAN functions. The grid ZONALMEDIAN function only operates on integer grids. **Slopegrid** is not an integer grid. The grid INT function is therefore used in conjunction with the grid ZONALMEDIAN function. Since the grid INT function rounds values to the nearest integer, **slopegrid** is multiplied by 1000 to reduce the degree of accuracy lost during the operation. This means **mediansgrid** actually stores the median slope times 1000. If necessary, previously existing **slopegrid**, **meansgrid** and **mediansgrid** grids are deleted using the arc KILL ALL command. Figure 4.5. lists the grid syntax.

```
slopegrid = slope (elevgrid,percentrise)
meansgrid = zonalmean (subgrid,slopegrid,#)
mediansgrid = zonalmedian (subgrid, INT (slopegrid * 1000),
#)
```

Figure 4.5. Grid Syntax for Determining Mean and Median Surface Slope.

Each polygon in **subcov**, except the universal one around the watershed perimeter, is then processed. The AML SHOW function is used along with the grid CELLVALUE command to determine the elevation (elevation grid) at the centroid and the mean (**meansgrid**) and median (**mediansgrid**) surface slope for each subbasin. The value from **mediansgrid** is divided by 1000. The elevation, mean slope and median slope values are written to **subcov.pat**//cenz, means and medians, respectively.

Stream Node Attributes: Each node in **hydrocov** is processed. The AML SHOW function is used along with the grid CELLVALUE command to determine the elevation of each node. The elevation is written to **hydrocov.nat**//nodez and an array (*nodez*(node)).

4.3.2.5. Establish Sources, Sinks and Subbasin Outlets.

'Clipped' Procedure: So far the identification of source, sink and subbasin elements depends on them being created by the intersection of **streamcov** with **subcov** or having been snapped to nodes in **subcov**. The system can also determine source, sink and subbasin outlet elements for the case that the stream coverage was clipped or intersected with the subbasin coverage before the program is run. This is specified with the *clipped* variable. Sinks, sources and subbasin outlets are established by looking at all the nodes in **hydrocov** and determining if there are actually lines from **subcov** at that location.

Each node in **hydrocov** is processed. The xy coordinates of the node are used along with the arcplot ASELECT command to attempt to select a line from **subcov**. Prior to the attempt the search tolerance is set to 10 with the arcplot SEARCHTOLERANCE command. If in fact a **subcov** line gets selected the **hydrocov** node is located at an intersection and is therefore a source, sink or subbasin outlet element. If that is the case the node gets marked by setting **hydrocov.nat**//hectype to 0, which is the same hectype as nodes created by the intersection.

Record Data and Create Selection File: Source, sink and subbasin outlet nodes are selected from **hydrocov**. The nodes are identified by having **hydrocov.nat**//hectype 0 or **hydrocov.nat**//snapped 1 (only checked if node snapping was actually done). The arcplot WRITESELECT command is used to write the selection set to a file and the **hydrocov.nat**//hectype is set to 1 for the selection set.

4.3.3. Establish Channel System.

Stream lines are classified into two general types. Stream lines can be part of the channel system that carries water from upstream features (channel system) or they can be tributaries to the channel system (non-channel system). Further, lines are distinguished based on being part of a reservoir defined by double-line connections between two stream nodes.

The channel system is defined by being downstream of hydrologic elements. There is always a source or a subbasin outlet at the upstream ends of the channel system. That means that the channel system can also be

defined by being downstream of sources, sinks and subbasin outlets. The system identifies the channel system by tracing downstream of the source, sink and subbasin outlet nodes established previously. Reservoir lines are identified by being enclosed polygons in the stream coverage.

All the nodes and lines in hydrocov are selected and the arcplot TRACE command is used along with the selection file specifying source, sink and subbasin outlet nodes to identify the nodes and lines downstream of the source, sink and subbasin outlet nodes. The arcplot TRACE command automatically writes the traced features to a selection file. The features in that selection file represent the channel system.

The channel system is selected using the arcplot READSELECT command and sources, sinks and subbasin outlets (**hydrocov.nat**//hectype 1) are removed from the selection set. The channel system nodes (except source, sink and subbasin outlet) and lines are marked by setting **hydrocov.nat**//hectype and **hydrocov.aat**//hectype to 4.

The selection set is switched and the sources, sinks and subbasin outlets are again removed from the selection. The non-channel system lines

and nodes are marked by setting **hydrocov.nat**//hectype and **hydrocov.aat**//hectype to 2.

4.3.4. Establish Channel Elements.

The hydrologic element represented by each node (seven element and hectype system) can now be identified by its relation to other elements. General ARC/INFO node, line and polygon topology is however insufficient for this purpose, because it does not directly provide information on how each node relates to connecting lines. That information is vital for determining the hydrologic element type of a node. For example: A node having multiple upstream lines is defined as a junction; a node having multiple downstream lines is defined as a diversion; etc.

Therefore, the ARC/INFO topology is expanded to include information on how each node relates to connecting lines. That information is stored as 'node type'.

Each node has two node types. One node type (rnodetype) is calculated taking into account all the lines in **hydrocov**. Another node type (hnodetype) is calculated taking into account only lines belonging to the channel system. Consider, for example, the subbasin in the upper right corner of Figure 3.3. When taking into account all the lines there are four ‘junctions’ whereas when taking into account only lines from the channel system there are only ‘interior’ nodes.

4.3.4.1. Create Hydrotopology Arrays.

The first step in building this extended topology is to create arrays storing the number of lines upstream and downstream of each node. These arrays are termed hydrotopology arrays. A line is defined as being upstream or downstream of a node by having that node as tnode or fnode respectively. In this step it is differentiated among (1) lines that are part of the channel system and those that are not, and (2) lines that are part of a reservoir and lines that are not. That information is stored in the arrays listed in Table 4.5.

Hydrotopology Array	**Description**
r#nodes(node)	number of lines that have this node as tnode or fnode.
r#fnodes(node)	number of lines that have this node as fnode.
r#tnodes(node)	number of lines that have this node as tnode.
r#rnodes(node)	number of reservoir lines that have this node as tnode or fnode.
r#rfnodes(node)	number of reservoir lines that have this node as fnode.
r#rtnodes(node)	number of reservoir lines that have this node as tnode.
h#nodes(node)	number of channel system lines that have this node as tnode or fnode.
h#fnodes(node)	number of channel system lines that have this node as fnode.
h#tnodes(node)	number of channel system lines that have this node as tnode.
h#rnodes(node)	number of channel system reservoir lines that have

	this node as tnode or fnode.
h#rfnodes(node)	number of channel system reservoir lines that have this node as fnode.
h#rtnodes(node)	number of channel system reservoir lines that have this node as tnode.

Table 4.5. Description of Hydrotopology Arrays.

To define the value of the arrays each line in **hydrocov** is processed. For the fnode 1 is added to the *r#nodes*(fnode) and *r#fnodes*(fnode) arrays. If the line is also a reservoir line, which it is if either the **hydrocov.aat**//rpoly or lpoly items are not 1 (the universal polygon), 1 is also added to the *r#rnodes*(fnode) and *r#rfnodes*(fnode) arrays. The same is done for tnode. If the line is part of the channel system (**hydrocov.nat**//hectype 1 or 4) the same procedure is also done for the *h#nodes*(node), *h#fnodes*(node), etc. arrays.

4.3.4.2. Calculate Node Types.

With the information from the previous step the node type can be calculated. Two node types are calculated: The r (real) node type which takes into account all lines and the h (hydro) node type which takes into account

only those lines that are part of the channel system. Each node in **hydrocov** is processed. The rnodetype is determined based on the value of the arrays determined in the previous step as shown in Table 4.6. Similarly the hnodetype is determined as shown in Table 4.7. The rnodetype and hnodetype of each node is also stored in the *rnodetype*(node) and *hnodetype*(node) arrays, respectively. The node type is stored in arrays, because they are generally easier to access than table values.

Rnodetype	Criteria
updangling	*r#nodes*(node) = 1 and *r#fnodes*(node) = 1
dndangling	*r#nodes*(node) = 1 and *r#tnodes*(node) = 1
interior	*r#fnodes*(node) = 1 and *r#tnodes*(node) = 1
junction	*r#fnodes*(node) = 1 and *r#tnodes*(node) >= 2
diversion	*r#tnodes*(node) = 1 and *r#fnodes*(node) >= 2
mreservoir	*r#rfnodes*(node) = 1 and *r#rtnodes*(node) = 1
upreservoir	*r#rfnodes*(node) = 2 and *r#rtnodes*(node) = 0
dnreservoir	*r#rtnodes*(node) = 2 and *r#rfnodes*(node) = 0
dreservoir	*r#rtnodes*(node) = 1 and *r#rfnodes*(node) = 1 and *r#fnodes*(node) = 2
jreservoir	*r#rtnodes*(node) = 1 and *r#rfnodes*(node) = 1 and *r#tnodes*(node) = 2

Table 4.6. Rnodetype Criteria.

Hnodetype	**Criteria**
updangling	*h#nodes*(node) = 1 and *h#fnodes*(node) = 1
dndangling	*h#nodes*(node) = 1 and *h#tnodes*(nodes) = 1
interior	*h#fnodes*(node) = 1 and *h#tnodes*(node) = 1
junction	*h#fnodes*(node) = 1 and *h#tnodes*(node) >= 2
diversion	*h#tnodes*(node) = 1 and *h#fnodes*(node) >= 2
mreservoir	*h#rfnodes*(node) = 1 and *h#rtnodes*(node) = 1
upreservoir	*h#rfnodes*(node) = 2 and *h#rtnodes*(node) = 0
dnreservoir	*h#rtnodes*(node) = 2 and *h#rfnodes*(node) = 0
dreservoir	*h#rtnodes*(node) = 1 and *h#rfnodes*(node) = 1 and *h#fnodes*(node) = 2
jreservoir	*h#rtnodes*(node) = 1 and *h#rfnodes*(node) = 1 and *h#tnodes*(node) = 2
none	*h#nodes*(node) = 0 and *h#rnodes*(node) = 0

Table 4.7. Hnodetype Criteria.

4.3.4.3 Classify Elements Based on Hectype System.

After the node type has been identified the nodes are further classified as elements based on the hectype system. This is done in the same loop as the previous step (Section 4.3.4.2). Each element's hectype is determined based on the criteria shown in Table 4.8.

Hectype	**Element Type**	**Criteria**
3	junctions	*hnodetype*(node) = junction and *hectype* <> 1
8	diversions	*hnodetype*(node) = diversion
6	sinks	*rnodetype*(node) = dndangling
7	sources	*rnodetype*(node) = updangling and *hnodetype*(node) = updangling
10	reservoir	*hnodetype*(node) = dnreservoir
1	subbasin outlet	**hydrocov.nat**//hectype = 1

Table 4.8. Channel Element Hectype Criteria.

Consider, for example, node number nine (9) in Figure 3.5. The hydrotopology arrays will indicate that the node is not part of a reservoir, *r#rnodes*(node), *r#rfnodes*(node), *r#rtnodes*(node), *h#rnodes*(node),

h#rfnodes(node), *h#rtnodes*(node) are zero. Further, the node is the from-node to two stream lines and to-node to one stream line, *r#nodes*(node) = 3, *r#fnodes*(node) = 2, *r#tnodes*(node) = 1, *h#nodes*(node) = 3, *h#fnodes*(node) = 2, *h#tnodes*(node) = 1. Based on the criteria presented in Tables 4.6 and 4.7 the r (real) and h (hydro) node type, *rnodetype*(node) and *hnodetype*(node) of the node is 'diversion'. This makes the node a 'diversion' hectype based on the criteria presented in Table 4.8.

The element is also assigned a hecid value from an incrementing *hecid* variable. The hecid and hectype values are written to **hydrocov.nat//**hecid and hectype, respectively. Again, because arrays are more easily accessed than tables, the hectype, hecid, hecx (**hydrocov.nat//**x-coord) and hecy (**hydrocov.nat**//y-coord) are written to the *hectype*(hecid), *hecx*(hecid), *hecy*(hecid), *nhectype*(node), *nhecid*(node) arrays.

At this point in the program the hydrotopology arrays have outlived their primary purpose. However, the *h#fnodes*(node) and *h#tnodes*(node) arrays will be of further use. They are used to store the number of downstream and upstream elements to the node respectively. For that purpose the *h#fnodes*(node) and *h#tnodes*(node) arrays are copied to arrays indexed on

hecid (*hh#fnodes*(hecid) and *hh#tnodes*(hecid)). The number of lines belonging to the channel system upstream and downstream of a node is however not always equal to the number of elements upstream or downstream of a node. A reservoir outlet has two reservoir lines upstream, which are also part of the channel system. These lines do not represent two upstream elements and the *hh#tnodes*(hecid) is therefore reduced by the value of the *h#rtnodes*(node) array.

4.3.4.4. Calculate Subbasin Outlet Node Information.

Now that the nodes representing subbasin outlets have been identified they are related to their respective subbasins. Each line in **hydrocov** is processed. If the hectype of the tnode is 1 or 6 the tnode is a possible outlet of a subbasin.

In case there are more than one possible outlets from the subbasin, as is the case if there is a diversion in the subbasin, a check has to be performed to see which one is the actual outlet. That is done by comparing the elevation of the nodes which is stored in the *nodez*(*node*) array. The node with the

lowest elevation is considered to be the outlet. The node number is stored in the *polydnnode*(poly) array.

The *hh#tnodes*(hecid) array is increased by 1 for the subbasin outlet. That is done to account for the fact that there is one more upstream element to the element than is indicated by the number of channel lines upstream of the node.

4.3.5. Calculate Connectivity.

4.3.5.1. Channel Elements and Reaches.

Channel elements are represented by nodes. They are connected by reaches which are represented by one or more lines of the channel system. Connectivity among the channel elements is calculated by looking at each node, and determining if it is an upstream end of a reach. If it is an upstream end of a reach, the downstream lines are traced until a downstream end of a reach is found. All the lines between the upstream and downstream ends are combined into one reach element. If a reservoir is encountered during the

tracing operation the reservoir lines are followed downstream until the reservoir element, located at the reservoir outlet, is found. If the upstream element is a diversion it will be processed several times.

Determine if Node is an Upstream Element: All nodes in **hydrocov** belonging to the channel system (**hydrocov.nat**//hectype <> 2) are processed. A check is done to determine if the node is an upstream end of a reach. Upstream elements of reaches are *nhectype*(node) 1, 3, 7, 8, 10 and reservoir diversions (*hnodetype*(node) = dreservoir). See the discussion on reservoir diversions later in this section for more details.

If the upstream element is a diversion (**hydrocov.nat**//hectype = 8) it is the upstream end of multiple reaches and will be processed several times as indicated by the *hh#fnodes*(hecid) array. Reservoirs are diversions if *hh#fnodes*(hecid) >= 2. If the node is a valid upstream element of a reach it is further processed, otherwise the next node is examined.

Trace Downstream until Downstream Element is Found: Each line in **hydrocov** belonging to the channel network (**hydrocov.aat**//hectype = 4) that has not been processed before (**hydrocov.aat**//hecid = 0) is processed to find

the line that has this node as fnode. That line is the first downstream line of the downstream reach. Once the line is found it is added to the current selection. Variables storing the length and slope of the reach (*heclength*, *hecslope*) are updated.

The line's tnode is evaluated to see wheter it is a valid downstream element of a reach. If the line's tnode is a valid element the end of the reach is found and the reach is 'closed', otherwise the next downstream arc has to be found. Valid downstream elements include **hydrocov.nat**//hectype 1, 3, 6, 8, 9 and the reservoir types 'jreservoir' and 'upreservoir' (*hnodetype*(node)). Note that with upreservoirs and jreservoirs the channel closes, but we have to continue downstream along the shore of the reservoir to find the dnreservoir.

Reservoir Extension: The 'reservoir extension' is similar to the previous step However, the variables *heclength* and *hecslope* do not get updated for each line. Also care has to be taken not to leave the reservoir on a reservoir diversion. Only lines that are adjacent to a reservoir (**hydrocov.aat**//rpoly# <> 1 or lpoly# <> 1) are therefore valid.

If the lines tnode is a reservoirs (**hydrocov.nat**//hectype = 10) the reach is closed. For each reservoir we encounter this way we have to update the *hh#tnodes*(hecid) array since there is now one more upstream elements than indicated by the number of channel system lines entering the node.

Record Reach Data: Once the reach is closed the **hydrocov.aat**//hecid item of all the lines in the reach gets updated with the current *hecid* value. The *hectype*(hecid), *hecx*(hecid), *hecy*(hecid) and *hecdnid*(n, hecid) [n is an integer indicating the specific downstream element, for the case of a reach always n = 1] variables get updated. The hecupid, hecdnid, heclength and hecslope values are written to all the lines of the reach element (**hydrocov.att**//hecupid, hecdnid, heclength and hecslope). Note that for the case of multiple hecupid's or hecdnid's, 0 will be recorded.

Record Upstream Element Data: The hecid of the reach is written to the **hydrocov.nat**//hecdnid item of the upstream element. In case of a diversion 0 will be recorded for **hydrocov.nat**//hecdnid. The *hectype*(hecid), *hecx*(hecid) and *hecdnid*(n, hecid) arrays are updated for the upstream element.

Record Downstream Element Data: The hecid of the reach is written to the **hydrocov.nat**//hecupid item of the downstream element. In case of a junction 0 will be recorded. The *hecupid*(n, hecid) array is updated for the downstream element. The *hectype*(hecid), *hecx*(hecid), *hecy*(hecid) and *hecupid*(n, hecid) arrays are updated for the downstream element.

4.3.5.2. Subbasins.

Subbasin elements are represented by polygons or their centroids. They are connected to subbasin outlets by means of a link. The connectivity among the subbasins and the subbasin outlets was established in Section 4.3.4.4. Here the subbasins get assigned hecids and several attributes are calculated and recorded.

Create Subbasin Point Coverage: In following attribute calculations a point coverage representing the subbasins is needed. A point coverage of the subbasin centroids (**subcov2**) is created from **subcov** using the arcplot CREATE command. A previously existing **subcov2** is deleted using the arc KILL ALL command.

Point topology is created using the arc BUILD command and the xy coordinates get added to **subcov2.pat** using the arc ADDXY command. The intx and inty items get added to **subcov2.pat** and **subcov.pat** and are assigned the x and y coordinates, respectively. The purpose of the intx and inty items is to preserve the original xy coordinates, because subsequent operations will modify the x-coord and y-coord values.

Set Up Hydrocov for Network and Dynamic Segmentation: **Hydrocov** is prepared for dynamic segmentation and network processing. Routes are created for each line in **hydrocov** using the arc ARCSECTION and MEASUREROUTE commands. **Hydrocov** is designated as network coverage using the arcplot NETCOVER command.

Record General Subbasin Data: Each polygon in **subcov**, except for the universal one, is processed. The current *hecid* is written to **subcov.pat**//*hecid*. **Subcov.pat**//*hectype* is set to 5. The number of the subbasin outlet node is obtained from *polydnnode*(poly). The hecid of the subbasin outlet, as obtained from *nhecid*(node), is written to **subcov.pat**//hecdnid. The *hectype*(hecid), *hecx*(hecid), *hecy*(hecid) and *hecdnid*(n, hecid) arrays are updated for the polygon. The *hecupid*(n, hecid) array for the subbasin outlet is updated.

Write Coordinates of Subbasin Boundary to a File: The coordinates of the subbasin boundary are written to a text file. This is done in the same loop as the previous step. All the lines belonging to the subbasin polygon are selected. The selection set is written to a file with the arcplot

WRITESELECT command. That selection file is then used with the arc RESELECT command to create a coverage consisting only of the lines of the subbasin polygon (**bound**). A previously existing **bound** is deleted using the arc KILL ALL command. Then the arc UNGENERATE command is used to write the coordinates of the subbasin boundary to a text file. The name of the text file is written to **subcov.pat**//b-coord.

Calculate Attributes of Longest Flow Path: The coordinates, elevation of upstream end and length of longest flowpath are computed. This is done in the same loop as the previous step.

All the **hydrocov** lines in the subbasin polygon are selected. What polygon the lines are in can be determined from **hydrocov.aat**//subcov#. All the **hydrocov** lines in the subbasin polygon are processed. For each line the fnode and tnode is added to the current **hydrocov** node selection set. All the **hydrocov** nodes in the subbasin polygon are then written to a selection file using the arcplot WRITESELECT command. This

The distance between all the nodes in the subbasin is computed and written to a file (**distfile**) using the arcplot NODEDISTANCE command. The

distfile records specifying distances from the subbasin outlet node (*polydnnode*(node)) are selected. Each of the selected records in **distfile** is then processed to find the record with the greatest node to node distance. That record specifies the length of the longest flow path and the node representing the upstream end of the longest flow path.

The elevation of the upstream end is obtained from the *nodez*(node) array. To get the coordinates of the longest flow path, the arcplot TRACE command is used to identify the lines downstream of the node. The coordinates of those lines are then written to a text file in the same manner the coordinates of the boundary are written to a file. The length, elevation of the upstream end, and the name of the file holding the coordinates of the longest flow path are written to the **subcov.pat**//flength, fupz and f-coord.

Calculate Length Along Longest Flow Path from Point Nearest to Subbasin Centroid to Subbasin Outlet: This is done in the same loop as the previous step. The coordinates of the point on the longest flowpath nearest to the centroid are found from the coverage of the longest flow path (previous step) and **subcov2** using the arc NEAR command. The intx and inty items are used to relate back to **subcov** and select the point belonging to the current

subbasin. The x-coord and y-coord items cannot be used, because their values are modified by the operation.

The length is obtained by adding two parts. The first part (flength2a) is the length from the next downstream node from the point to the subbasin outlet. Flength2a is determined from **distfile**. The second part (flength2b) is the fractional length along the line obtained with dynamic segmentation. Figure 4.6 illustrates this concept. The length is written to **subcov**//flength2. Figure 4.7. lists the arcplot syntax.

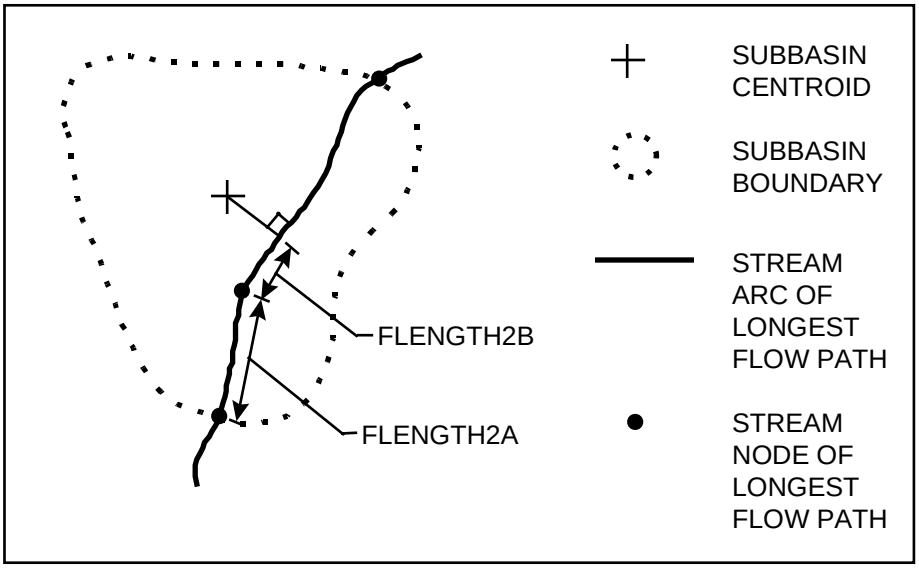

Figure 4.6. Flength2 Illustration.

```
&sys arc near subcov2 path line 1000000 # location
unselect subcov2 points
aselect subcov2 points ( intx = %:cur6070.intx% )
unselect subcov2 points ( inty ne %:cur6070.inty% )
&sv nearx = [show select subcov2 point 1 item x-coord]
&sv neary = [show select subcov2 point 1 item y-coord]
unselect hydrocov arcs
aselect hydrocov arcs one *
%nearx%, %neary%
&sv tnode = [show select hydrocov line 1 item tnode#]
unselect distfile info ( hydrocov#b ne %tnode% )
&sv flength2a = [show select distfile info 1 item network]
measure route hydrocov fpath
%nearx%, %neary%
&sv flength2b = [extract 5 [show measure route]]
&sv flength2 = %flength2a% + %flength2b%
&s :cur6070.flength2 = %flength2%
```

Figure 4.7. Arcplot Flength2 Syntax.

4.3.5.3. Reservoir Diversions.

Reservoir diversions are represented by nodes on reservoirs (see node type ‘dreservoir’ in Figure 3.4). Similar to the channel elements described in Section 4.3.5.1 they are connected to downstream channel elements by reaches. Their downstream connectivity was calculated in Section 4.3.5.1. However, the seven element system does not support reservoir diversions as separate elements. In that system a diversion out of a reservoir is handled by having multiple reaches downstream of the reservoir element. The reservoir

diversion therefore has to be eliminated by assigning the corresponding reservoir to the upstream end of the reach.

Find Reservoir Diversion: Each node in **hydrocov** belonging to the channel network and not having been classified as another hectype is processed. If the node is *hnodetype*(node) dreservoir the node is further processed. Otherwise the search is continued.

Find Downstream Reach: The hecid of the downstream reach is obtained by processing each line in **hydrocov**. The line is the first line of the downstream reach if it has the node as fnode and if it is not located along the reservoir (**hydrocov.aat**//rpoly# = 1 and lpoly# = 1).

Find Reservoir Element: The dnreservoir of the reservoir is found by first finding the upstream line to the reservoir diversion in a manner similar to how the downstream line was found in the previous section. That line is determied by having the node as tnode. Then the **hydrocov** polygon number of the reservoir is determined. It is the non-1 value of **hydrocov.aat**//rpoly# and lpoly#. Knowing the polygon number of the reservoir all the lines in **hydrocov** belonging to the reservoir are selected. Each of the selected lines is

then processed. The dnreservoir is the first fnode or tnode encountered that has *hnodetype*(node) dnreservoir.

Record Data: Once all the elements involved are determined, several arrays are updated. Any dnreservoir encountered this way has one more downstream element as indicated by the number of downstream lines (or as previously assumed). The *hh#fnodes*(hecid) arrays is therefore increased by 1. The hecid of the reservoir is written to **hydrocov.aat**//hecupid and *hecupid*(n, hecid) array.

4.3.6. Create Symbolic Coverage.

A symbolic coverage representing the watershed in a stick diagram fashion is produced. The creation of the coverage is based solely upon data collected throughout the previous program run.

An empty coverage (**symcov**) is created from **hydrocov** using the arc CREATE command. A previously existing **symcov** is deleted. Node

topology is created and the hecid item is added to **symcov.nat**. Line topology is created and the hecid, uphecid and dnhecid items are added to **symcov.aat**.

4.3.6.1. Channel System.

The channel system is created by processing each element previously established. If the element is an upstream end of a reach it has a corresponding reach element. That reach element has a corresponding downstream element. A line will be added from the coordinates of the upstream element to the coordinates of the downstream element. Then the next element is processed.

Find Upstream Element: Each element is processed based on the number of hecid's assigned. The element is an upstream end of a reach if its *hectype*(hecid) is 1, 3, 7, 8 or 10. Diversions (hectype 8) and reservoirs with multiple downstream elements (*hh#fnodes*(hecid) > 1) have to be processed multiple times.

Find Reach: The first downstream element is determined based on the hecdnid of the current element. That element is a reach.

Find Downstream Element: The second downstream element is determined based on the hecdnid of the reach.

Add Line: A line is added to **symcov** based on the xy coordinates of the upstream and downstream elements. The hecid of the reach is written to **symcov.aat**//hecid item. The hecid of the upstream and downstream elements of the reach is written to **symcov.aat**//uphecid and dnhecid elements, respectively.

4.3.6.2. Subbasins.

Subbasins are processed similar to the Channel System in Section 4.3.6.1 except that there is no element between the subbasin and the subbasin outlet. Each element is processed. If it is a subbasin, its corresponding subbasin outlet is found and a line is added between them.

Find Subbasin Element: Each element is processed based on the number of hecid's assigned, similar to the previous step, starting with hecid 1. A line will be added from the subbasin centroid to the subbasin outlet if in fact we are at a subbasin element (*hectype*(hecid) = 5).

Find Subbasin Outlet Element: The subbasin outlet is found based on the *hecdnid*(hecid) of the subbasin element.

Add Line: A line is added based on the xy coordinates of the subbasin element and subbasin outlet element. The hecid item is assigned 0.

4.3.6.3. Update Symcov.nat.

A link will have to be made from the nodes in symcov to other files based on a common hecid. Since so far the hecid's are only written to **symcov.aat**, they have to be transfered to **symcov.nat**. First node and line topology is build.

Each line in symcov is processed. The hecid of the upstream and downstream ends is writen to a new array called *hecid2*(node). Each node in symcov is processed. The *hecid2*(node) array is used to write the hecid value of the element to the hecid item in **symcov.nat**.

4.3.7. Relate Attributes to Symbolic Coverage.

The symbolic coverage does not contain any attribute values. Attributes are stored in the subbasin coverage and the hydrologic coverage and can be accessed through relates. The attributes that can be accessed this way are not limited to the ones from this discussion. Any attributes from the input coverages can be accessed this way.

Three relates are established using the arc RELATE ADD command. The relates are:

- **symcov.aat** to **hydrocov.aat**
- **symcov.nat** to **hydrocov.nat**
- **symcov.nat** to **subcov.pat**

The relates are based on a common hecid value included in the corresponding records of each pair of tables.

4.3.8. Create Output.

At this point the program has finished compiling the data. Part of the final output, the hydrologic and symbolic coverages, is complete. This part of the program creates the DXF file, general output file and HEC-HMS basin file.

4.3.8.1. Create DXF File.

The arc ARCDXF command is used to create a DXF file of the streams and subbasins in the watershed. The command can only convert a single coverage to a DXF file. The coverages **streamcov** and **subcov** have to be combined into a single one. Further, their attribute fields have to be consistent and contain the 'dxf-layer' item if the features are to be on separate layers.

Create Combined Coverage: **Streamcov** and **subcov** are copied to **streamcov2** and **subcov2**, respectively with the arc COPY command. **Streamcov2.aat** and **subcov2.aat** are then deleted with the tables ERASE command. Line topology is rebuilt with the arc BUILD command and the 'layers' item is added to **streamcov2.aat** and **subcov2.aat** using the arc ADDDITEMS command. The 'layers' item in **streamcov2.aat** and **subcov2.aat** is then set to 'streams' and subbasins, respectively, using the arcplot CALCULATE command. The coverages are then combined into a single coverage (**dxfcov**) using the arc APPEND command.

Create DXF File: Finally the arc ARCDXF command is used to create the DXF file from **dxfcov**. The temporary coverages **streamcov2**, **subcov2** and **dxfcov** are deleted using the arc KILL ALL command.

4.3.8.2. Create General Text File and HEC-HMS Basin File.

The general text and HEC-HMS basin files list hydrologic elements and their attributes by element type (see Section 3.4.7). Since the files are structured approximately the same they are created at the same time. The attribute data is accessed from the hydrologic and symbolic coverages through relates.

Two output files are opened with the AML OPEN function and headers are written to the file using the AML WRITE function. Each element type is processed seperately and the corresponding information is written to the text file based on the value in the related attribute file as show in Table 4.9.

Element Type	Information Written to File	Description
Subbasin	hecid	ID
	x-coord	x coordinate of centroid
	y-coord	y coordinate of centroid
	cenz	elevation of centroid
	b-coord	name of file holding coordinates of subbasin boundary
	f-coord	name of file holding coordinates of longest flow path
	fupz	elevation of upstream end of longest flow path.
	flength	length of longest flow path
	flength2	length along longest flow path from point nearest to subbasin centroid to subbasin outlet.
	area	area
	means	mean surface slope
	medians	median surface slope
	hecdnid	ID of downstream element.
Source	hecid	ID
	x-coord	x coordinate
	y-coord	y coordinate
	nodez	elevation
	hecdnid	ID of downstream element.
Reaches	hecid	ID
	heclength	length
	hecslope	slope
	hecupid	ID of upstream element.
	hecdnid	ID of downstream element.

Table 4.9. Description of Data Written to Output Files
(continued on next page).

Element Type	Information Written to File	Description
Junctions	hecid x-coord y-coord nodez hecupid(s) hecdnid	ID x coordinate y coordinate elevation ID of upstream element(s) ID of downstream element
Reservoirs	hecid x-coord y-coord nodez hecupid(s) hecdnid(s)	ID x coordinate y coordinate elevation ID of upstream element(s) ID of downstream element(s)
Diversions	hecid x-coord y-coord nodez hecupid hecdnids	ID x coordinate y coordinate elevation ID of upstream element ID of downstream element(s)
Sinks	hecid x-coord y-coord nodez hecupid(s)	ID x coordinate y coordinate elevation ID of upstream element(s)

Table 4.9. Description of Data Written to Output Files
(continued from previous page).

Element Type	Information Written to File	Source of Information
Subbasin	hecid	symcov.nat
	x-coord	subcov.pat
	y-coord	subcov.pat
	cenz	subcov.pat
	b-coord	subcov.pat
	f-coord	subcov.pat
	fupz	subcov.pat
	flength	subcov.pat
	flength2	subcov.pat
	area	subcov.pat
	means	subcov.pat
	medians	subcov.pat
	hecdnid	subcov.pat
Source	hecid	symcov.nat
	x-coord	hydrocov.nat
	y-coord	hydrocov.nat
	nodez	hydrocov.nat
	hecdnid	hydrocov.nat
Reaches	hecid	symcov.aat
	heclength	hydrocov.aat
	hecslope	hydrocov.aat
	hecupid	hydrocov.aat
	hecdnid	hydrocov.aat

Table 4.10. Source of Data Written to Output Files (continued on next page).

Element Type	Information Written to File	Source of Information
Junctions	hecid x-coord y-coord nodez hecupid(s) hecdnid	symcov.nat hydrocov.nat hydrocov.nat hydrocov.nat *hecupid*(n, hecid) hydrocov.nat
Reservoirs	hecid x-coord y-coord nodez hecupid(s) hecdnid(s)	symcov.nat hydrocov.nat hydrocov.nat hydrocov.nat *hecupid*(n, hecid) *hecdnid*(n, hecid)
Diversions	hecid x-coord y-coord nodez hecupid hecdnids	symcov.nat hydrocov.nat hydrocov.nat hydrocov.nat hydrocov.nat *hecdnid*(n, hecid)
Sinks	hecid x-coord y-coord nodez hecupid(s)	symcov.nat hydrocov.nat hydrocov.nat hydrocov.nat *hecupid*(n, hecid)

Table 4.10. Source of Data Written to Output Files
(continued from previous page).

After all the attributes have been written to the file it, is closed using the AML CLOSE function. An example general output file is presented in Appendix C.

4.4. Output Data Description.

This section describes the output data in detail. The output data contains of a HEC-HMS basin file, a general text file, a DXF file and a 'hydrologic' and a symbolic coverage.

4.4.1. HEC-HMS Basin File Output.

See Section 3.4.7 for a description of the HEC-HMS Basin File.

4.4.2. General Text File Output.

The general text file is in ASCII format. The file lists the hydrologic elements and their attributes by element type. The data written to the output file is listed in Table 4.11. An example output file is presented in Appendix C.

Element	Attributes
Subbasin	ID, coordinates of centroid, elevation of centroid, coordinates of subbasin boundary, coordinates of longest flow path, elevation of upstream end of longest flow path, length of longest flow path, length along longest flow path from point nearest to subbasin centroid to subbasin outlet, area, mean surface slope, median surface slope, ID of downstream element.
Source	ID, coordinates, elevation, ID of downstream element.
Reaches	ID, length, slope, ID of upstream element, ID of downstream element.
Junctions	ID, coordinates, elevation, ID of upstream element(s), ID of downstream element.
Reservoirs	ID, coordinates, elevation, ID of upstream element(s), ID of downstream element(s).
Diversions	ID, coordinates, elevation, ID of upstream element, ID of downstream element(s).
Sinks	ID, coordinates, elevation, ID of upstream element(s).

Table 4.11. Description of Data Written to General Text File Output.

4.4.3. DXF File Output.

The system can create an AutoCAD Drawing Exchange File (DXF) of the streams and the subbasins. The file is in ASCII format. It contains the

‘streams’ and ‘subbasins’ layers. The beginning of a DXF file is presented in Figure 4.8.

```
0
SECTION
  2
HEADER
  9
$EXTMIN
 10
74724.17
 20
1376741.13
  9
$EXTMAX
 10
177024.17
 20
1484841.13
  9
$LUPREC
 70
    14
  0
ENDSEC
  0
SECTION
  2
TABLES
  0
ENDSEC
  0
SECTION
  2
BLOCKS
  0
ENDSEC
  0
SECTION
  2
ENTITIES
  0
POLYLINE
  8
streams
 66
.
.
.
```

Figure 4.8. Beginning of a DXF File Output.

4.4.4. Hydrologic Coverage Output.

A ‘hydrologic’ coverage (**hydrocov**) is created. The main purpose of the coverage is to serve as a working coverage to store data and perform calculations during the program run. However, the coverage is also useful for trouble shooting purposes. See the discussion on viewing results in Section 4.5.5.

Hydrocov is an ARC/INFO line coverage of the streams. The coverage is the same as input stream coverage except that it has nodes at the intersections with the lines of the subbasin coverage. It contains node, line and polygon topology. Figure 4.9 shows an example of a **hydrocov** and Table 4.12 lists and describes the attribute data structure.

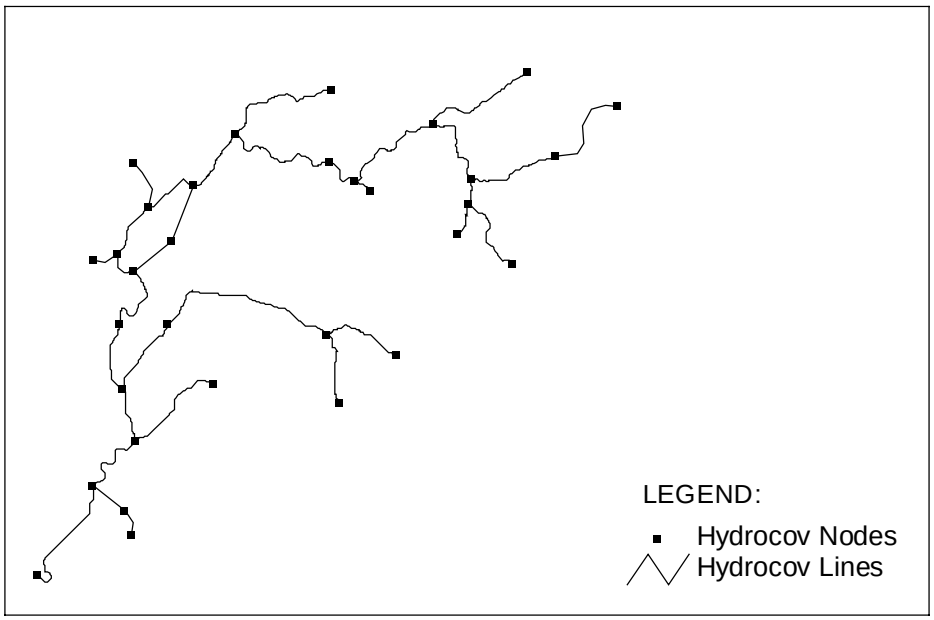

Figure 4.9. An Example Hydrologic Coverage for the Modified Tenkiller Watershed.

Attribute Table	Item	Description
hydrocov.nat	arc#	# of one connected line
	hydrocov#	# of node
	hydrocov-id	id of node
	x-coord	x coordinate of node*
	y-coord	y coordinate of node*
	hectype	feature hectype
	hecid	unique identification number
	hecupid	hecid of upstream element
	hecdnid	hecid of downstream element
	nodez	node surface elevation*
hydrocov.pat	area	area of polygon*
	perimeter	perimeter of polygon*
	hydrocov#	# of polygon
	hydrocov-id	id of polygon

* Units are the same as those in the input data files.

Table 4.12. Hydrologic Coverage Attribute Items (continued on next page).

Attribute Table	Item	Description
hydrocov.aat	fnode#	# of from-node
	tnode#	# of to-node
	lpoly#	# of left-polygon
	rpoly#	# of right-polygon
	length	length of line*
	hydrocov#	# of line
	hydrocov-id	id of line
	streamcov#	# of corresp. line in stream coverage
	streamcov-id	id of corresp. line in stream coverage
	hectype	feature hectype
	hecid	unique identification number
	hecupid	hecid of upstream element
	hecdnid	hecid of downstream element
	heclength	length of reach*
	hecslope	slope of reach*
	subcov#	# of corresponding polygon in subbasin coverage
	area	area of corresponding polygon in subbasin coverage*
	perimeter	perimeter of corresponding polygon in subbasin coverage*
	subcov-id	id of corresponding polygon in subbasin coverage
	cenz	not valued
	means	not valued
	medians	not valued
	b-coord	not valued
	f-coord	not valued
	fupz	not valued
	flength	not valued
	flength2	not valued

* Units are the same as those in the input data files.

Table 4.12. Hydrologic Coverage Attribute Items
(continued from previous page).

4.4.5. Symbolic Coverage Output.

A symbolic coverage (**symcov**) is created. The main purpose of the coverage is for the user to check the results of the system. See the discussion on viewing results in Section 4.5.5.

Symcov is an ARC/INFO line coverage. The lines in the coverage either represent reaches or links. The nodes in the coverage represent channel elements. The coverage contains node and line topology. Figure 4.10 shows an example of a **symcov** and Table 4.13 lists and describes the attribute data structure.

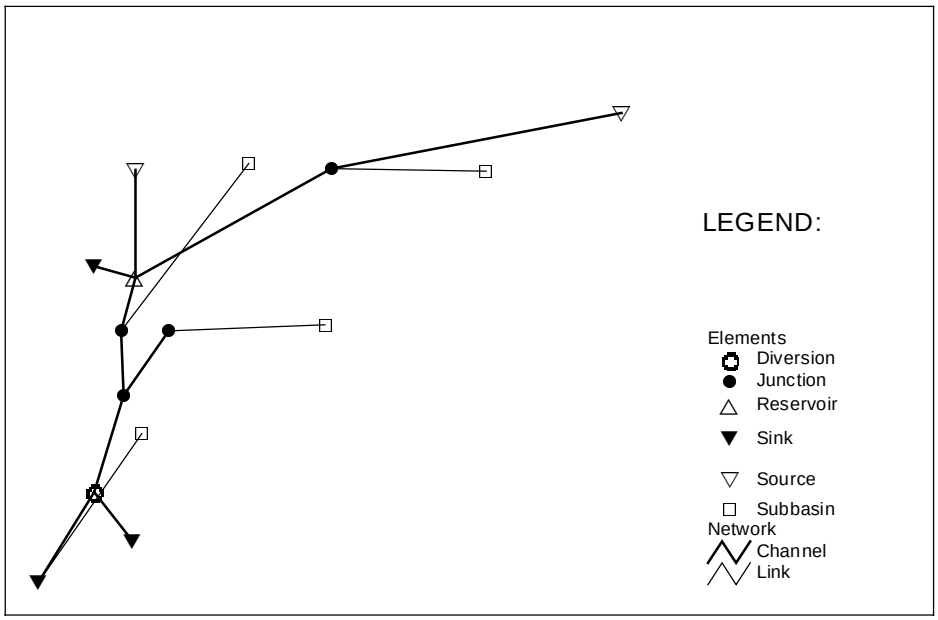

Figure 4.10. An Example Symbolic Coverage for the Modified Tenkiller Watershed.

Attribute Table	Item	Description
symcov.nat	arc#	# of one connected line
	symcov#	# of node
	symcov-id	id of node
	hecid	unique identification number
symcov.aat	fnode#	# of from-node
	tnode#	# of to-node
	lpoly#	# of left-polygon
	rpoly#	# of right-polygon
	length	length of line*
	symcov#	# of line
	symcov-id	id of line
	hecid	unique identification number
	uphecid	hecid of from-node
	dnhecid	hecid of to-node

* Units are the same as those in the input data files.

Table 4.13. Symbolic Coverage Attribute Items.

4.5. Using the HEC-PREPRO System.

4.5.1. Getting the System.

The system consist of AML programs that work inside ARC/INFO. The ARC/INFO Grid and Network modules are required. The systems consists of the following five AML programs:

- Main Program (hecprepro.aml).
- Pause Utility (hecpause.aml).
- Shell Program (hecshell.aml).
- Hydrocov Display Program (hechydro.aml).
- Symcov Display Program (hecsym.aml).

The programs can be downloaded via anonymous ftp as described in Table 4.14.

Data	**Value**
Site	ftp.crwr.utexas.edu
Login	anonymous
Password	*your e-mail address*
Directory	/pub/crwr/gishydro/hecprepro/
Mode	ASCII
Files	hecprepro.aml hecpause.aml hecshell.aml hechydro.aml hecsym.aml

Table 4.14. System Download Information.

4.5.2. Starting the System.

The main program is called hecprepro.aml. It can be started from the arc command line using the AML &RUN directive. The names of the three input data sets have to be specified at the command line. Other optional input parameters, like the user observation level, can also be specified at the command line. Input parameters are described in detail in the next sections. If the optional input parameters are not specified they are assigned default values. Figure 4.11 shows the syntax for starting the system with input stream coverage 'mystreams', input subbasin coverage 'mysubs', input elevation grid 'myelev' and user observation level 3. See Section 4.5.3 for a description of the user observation level. Table 4.15 summarizes the command line input. A more detailed discussion of the purpose of the individual variables follows later in this section.

```
Arc: &run hecprepro.aml mystreams mysubs myelev 3
```

Figure 4.11. Starting the System from the Command Line Syntax.

Variable Name	Description	Required	Default
streamcov	Stream Line coverage	Yes	--
subcov	Subbasin Polygon Coverage	Yes	--
elevgrid	Elevation Grid	Yes	--
*.oblevel**	User Observation Level	No	2
attrib	Attribute Collection?	No	Yes
clipped	Prior Clipping?	No	Yes
snapit	Node Snapping?	No	No
snaptol	Snapping Tolerance Distance	No	0
outfile	Name of Output File	No	hecprepro.out
dxffile	Create DXF File	No	No
dxfname	Name of DXF File	No	dxfout.dxf
hmsfile	Create HEC-HMS Basin File	No	No
hmsname	Name of HEC-HMS Basin File	No	hmsbasin.txt

* a period at the beginning of a variable name specifies that the variable is global.

Table 4.15. Command Line Input.

To facilitate the usage of the system a shell program called hecshell.aml is available. Hecshell.aml interactively prompts the user for the value of each individual variable. With hecshell.aml the user also has the option to run the system with the previous inputs or enter new inputs. If the user decides to enter new inputs the previous inputs will appear as defaults. Figure 4.12. shows part of the syntax for starting the system with hecshell.aml.

```
Arc: &run hecshell.aml
HECPREPRO:
HECPREPRO: ----------------------
HECPREPRO: ----------------------
HECPREPRO: --- HECPREPRO SHELL ---
HECPREPRO: ----------------------
HECPREPRO: ----------------------
HECPREPRO:
HECPREPRO:
HECPREPRO: ---------------------------------
HECPREPRO: --- Same Data as Previous Run? ---
HECPREPRO: ---------------------------------
HECPREPRO:
HECPREPRO: Description:
HECPREPRO: The program can be run with the same input as the
HECPREPRO: previous run.  The previous input variables were:
HECPREPRO:
HECPREPRO: Stream Coverage: tkst2
HECPREPRO: Subbasin Coverage: tksub
HECPREPRO: Elevation Grid: tkelev
HECPREPRO: User Observation Level: 0
HECPREPRO: Attribute Collection? Yes
HECPREPRO: Clipped? No
HECPREPRO: Node Snapping? No
HECPREPRO: Snapping Tolerance Distance: 0
HECPREPRO: Name of General Text File: hecprepro.out
HECPREPRO: DXF File? No
HECPREPRO: Name of DXF File: dxfout.dxf
HECPREPRO: HMS Basin File? Yes
HECPREPRO: Name of HMS Basin File: hmsbasin.txt
HECPREPRO:
HECPREPRO: Valid Responses:
HECPREPRO: Y - same data will be used
HECPREPRO: N - same data will not be used (default)
HECPREPRO:
HECPREPRO: Same Data as Previous Run?: n
HECPREPRO:

...
```

Figure 4.12. Starting the System with Hecshell.aml Syntax
(continued on next page).

```
...

HECPREPRO:
HECPREPRO: -----------------------
HECPREPRO: --- Stream Coverage ---
HECPREPRO: -----------------------
HECPREPRO:
HECPREPRO: Description:
HECPREPRO: The stream coverage has to have all the arcs
pointing
HECPREPRO: downstream.  Polygons are considered reservoirs.
HECPREPRO:
HECPREPRO: Valid Responses:
HECPREPRO: The name of a stream coverage
HECPREPRO:
HECPREPRO: Stream Coverage <tkst2>: tkst1
HECPREPRO:
HECPREPRO: -------------------------
HECPREPRO: --- Subbasin Coverage ---
HECPREPRO: -------------------------
HECPREPRO:
HECPREPRO: Description:

...

HECPREPRO:
HECPREPRO: Starting HECPREPRO ...
HECPREPRO:
HECPREPRO:
HECPREPRO: -----------------------
HECPREPRO: --- HECPREPRO START ---
HECPREPRO: -----------------------
HECPREPRO:

...
```

Figure 4.12. Starting the System with Hecshell.aml Syntax
(continued from previous page).

4.5.3. User Observation Level.

The system can be run with different user observation levels, which control how often the program will pause for user observation. Table 4.16. lists the possible user observation levels.

User Observation Level	**Description**
General	
0	no pause and no graphic display (for background running)
1	pause at error
2	pause at error and legends (default)
3	first level debug
4	second level debug
Special	
0.20, 0.30, etc.	makes first pause at Step 2, Step 3, etc.
9	quits instantly (entered at pause only)

Table 4.16. User Observation Levels.

When the program pauses for user observation the user can continue by pressing 'RETURN' or entering a new user observation level. If the user enters a '9' the program will quit instantly. Figure 4.13 shows an example of

a user observation level pause and a subsequent change to a user observation level 2.

```
.
.
.
Killed MEDIANSGRID with the ALL option
HECPREPRO:
HECPREPRO: Calculating subbasin median slopes
HECPREPRO:
Running... 100%
Percentage of Cells Processed: 100%
Feature cursor CUR1020 now declared using file
 SUBCOV.PAT with Read Write access
Feature cursor CUR1020 now opened with
5 reselected records out of 5
Fetched record 1 for Feature cursor CUR1020
Fetched record 2 for Feature cursor CUR1020
HECPREPRO:
HECPREPRO: Recording CENZ, MEANS and MEDIANS for subbasin
HECPREPRO: POLY: 2
HECPREPRO: CENZ: 365
HECPREPRO: MEANS: 3.17241
HECPREPRO: MEDIANS: 1.677
HECPREPRO:
HECPREPRO:<Return> or new observation level to continue: 2
.
.
.
```

Figure 4.13. User Observation Level Pause Example.

4.5.4. Advanced Options.

There are several advanced options as discussed below.

4.5.4.1. Attribute Collection Option.

The program structure is based on establishing hydrologic elements and connectivity among them. It is therefore possible to run the system without collecting attributes. That means the program will bypass operations not needed to establish hydrologic elements and connectivity among them. For example, for each hydrologic element the general text file will only contain ID, ID of upstream element(s) and ID of downstream element(s). Other data as shown in Table 4.11 is not written to the file.

Most errors in the input data produce significant errors when establishing connectivity among elements. Not collecting attributes might be desirable for the first system run when such errors are most frequently encountered

4.5.4.2. Clipped Option.

It is possible that the subbasin coverage was used to clip the stream coverage from a larger stream coverage. This requires a more thorough procedure for establishing source, sink and subbasin outlet elements. See Section 4.3.2.5 for detailed information.

The more thorough procedure is computationally more intensive and therefore increases program running time. However, if a doubt exists about the input data it should be chosen.

4.5.4.3. Node Snapping Option.

In Section 4.2 it was mentioned that the stream and subbasin coverages had to accurately represent the hydrologic properties of the area. This condition applies on every scale, all the way down to the fuzzy tolerance of the coverages. Consider for example Figure 4.14. There Grid routines were used to delineate subbasin boundaries from an intersection in the stream coverage.

However, the relatively coarse grid cell size caused the subbasin boundaries to be slightly offset from the stream junction. When constructing a schematic data model as discussed in Section 3.4 the result would be as shown in Figure 4.14. Obviously this was not the intent.

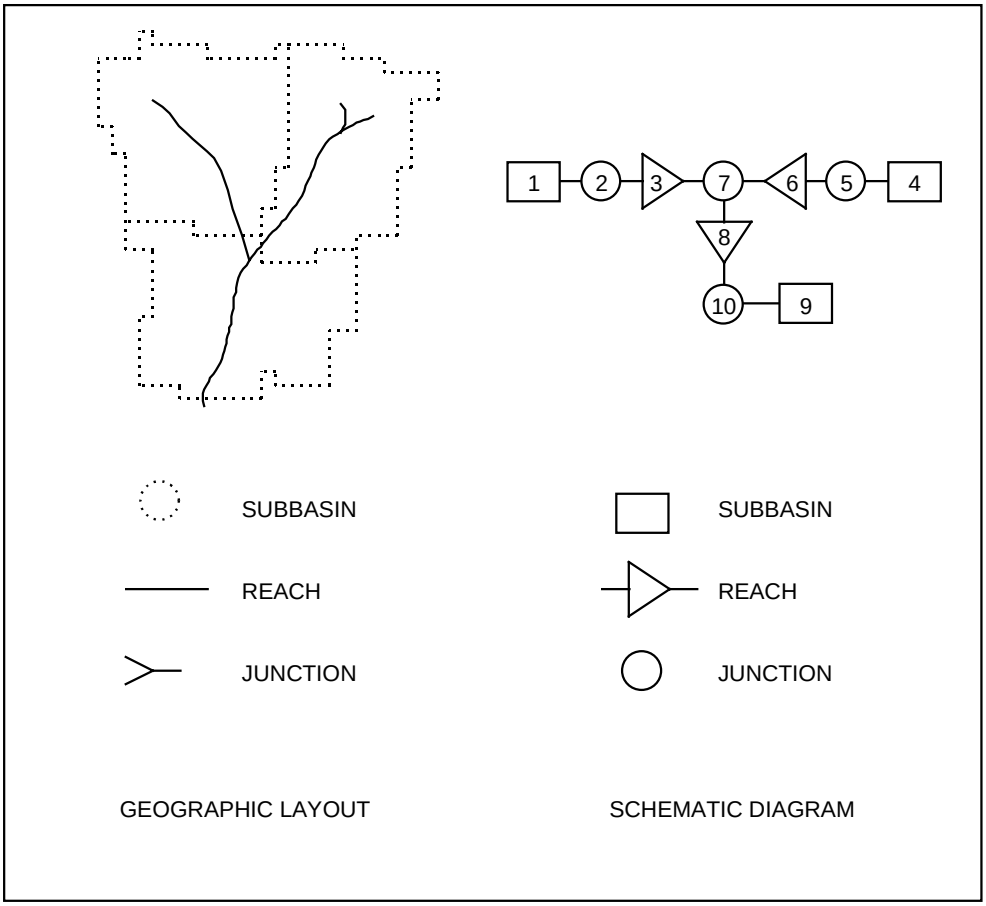

Figure 4.14. Example Input Data where Node Snapping is Applicable.

To correct this problem, the nodes of the stream coverage (e.g. junctions) can be snapped to nodes of the subbasin coverage before the schematic data model is constructed. Figure 4.15 shows the same watershed and resulting schematic data model after node snapping was performed.

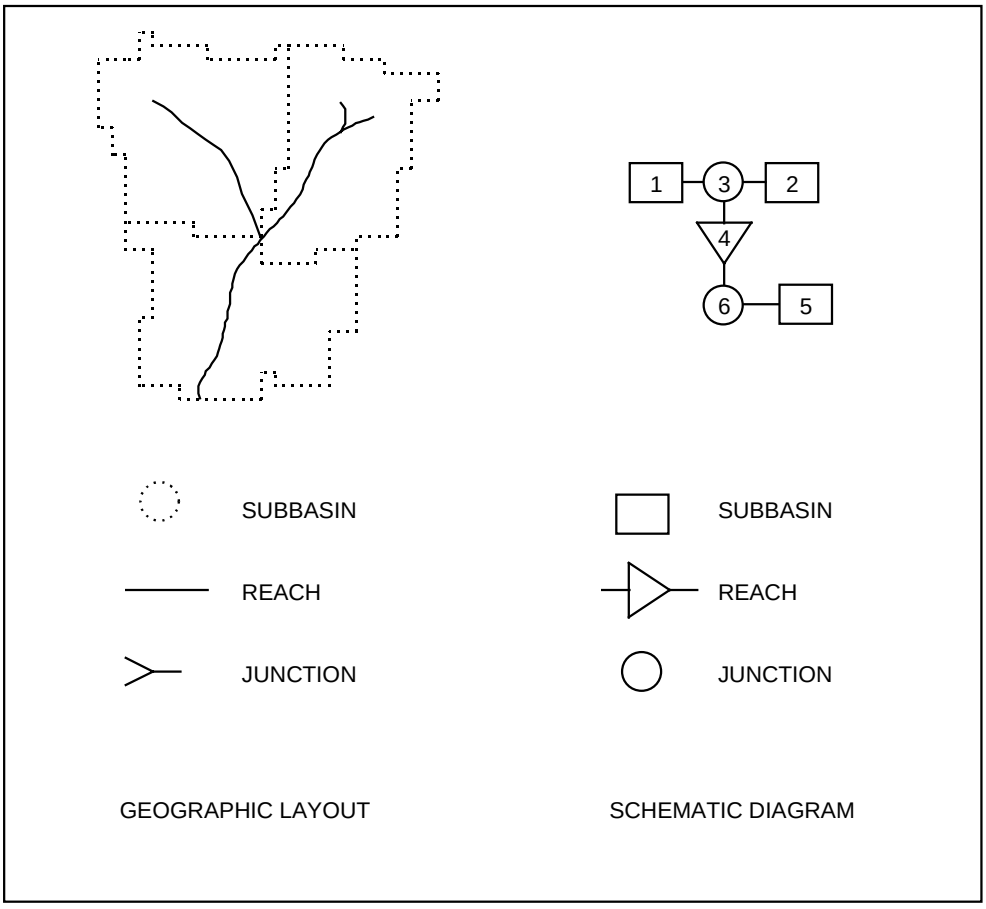

Figure 4.15. Example Input Data after Node Snapping.

When node snapping is to be performed it is required to enter a snapping tolerance distance, which specifies the maximum distance used for node snapping. If the discrepancies in the input data are due to delineation with Grid routines it is recommended that a tolerance of three times the grid cell size is used.

4.5.5. Viewing Output.

4.5.5.1. General Text File, HEC-HMS Basin File and DXF File.

The general text file, HEC-HMS basin file and DXF file are in ASCII format and can be viewed with any text editor or word processor.

4.5.5.2. Hydrologic Coverage.

The hydrologic coverage (**hydrocov**) can be displayed along with the subbasin coverage with the hechydro.aml program. The program will mark different element types with different symbols and display a legend in the command line window. The legend is presented in Figure 4.16.

```
HECPREPRO:
HECPREPRO: HYDROCOV LEGEND:
HECPREPRO: ------------------------------------------------
HECPREPRO:
HECPREPRO: LINES:
HECPREPRO: White:  Subbasin Boundary and Streams Outside
HECPREPRO:         Watershed Boundary
HECPREPRO: Yellow: Channel System
HECPREPRO: Red:    Non Channel System
HECPREPRO: Cyan:   Reservoir Shoreline
HECPREPRO:
HECPREPRO: SYMBOLS:
HECPREPRO: Green Triangles:           Subbasin Outlets
HECPREPRO: Red Triangles:             Sinks
HECPREPRO: Blue Triangles:            Sources
HECPREPRO: Green Circles:             Junctions
HECPREPRO: Blue Circles:              Diversions
HECPREPRO: Green Boxes:               Subbasins
HECPREPRO: Upsidedown Green Triangle: Reservoirs
HECPREPRO: Blue Dot:   Upstream End of Longest Flow Path
HECPREPRO: Yellow Box: Point on Longest Flow Path closest to
HECPREPRO:             Subbasin Centroid.
HECPREPRO:
HECPREPRO: TEXT:
HECPREPRO: White:  HECID of Channel Elements
HECPREPRO: Yellow: HECID of Reaches
HECPREPRO: Cyan:   HECDNID of Elements
HECPREPRO:
HECPREPRO: ------------------------------------------------
HECPREPRO:
```

Figure 4.16. Hydrologic Coverage Legend.

This hechydro.aml program is useful for troubleshooting **hydrocov**. The user can zoom into an area of interest using the arcplot MAPEXTEND command. **Hydrocov** is then easily redisplayed by running the hechydro.aml with the arcplot &RUN directive.

4.5.5.3. Symbolic Coverage.

The symbolic coverage (**symcov**) can be displayed along with the subbasin coverage with the hecsym.aml program. The program works similarly to hechydro.aml. It marks different element types with different symbols and display a legend in the command line window. The legend is presented in Figure 4.17.

```
HECPREPRO:
HECPREPRO: SYMCOV LEGEND:
HECPREPRO: ----------------------------------------
HECPREPRO:
HECPREPRO: LINES:
HECPREPRO: White:  Input Stream and Subbasin Lines
HECPREPRO: Green:  Reaches
HECPREPRO: Yellow: Links
HECPREPRO:
HECPREPRO: SYMBOLS:
HECPREPRO: Squares:              Subbasins
HECPREPRO: Stars:                Sources
HECPREPRO: Triangles:            Junctions
HECPREPRO: Upsidedown Triangles: Reservoirs
HECPREPRO: Circles:              Diversions
HECPREPRO: Diamonds:             Sinks
HECPREPRO:
HECPREPRO: TEXT:
HECPREPRO: Yellow: HECID of Elements
HECPREPRO:
HECPREPRO: ----------------------------------------
HECPREPRO:
HECPREPRO: General Text File: hecprepro.out
HECPREPRO: DXF File:          dxfout.dxf
HECPREPRO:
HECPREPRO: ----------------------------------------
HECPREPRO:
```

Figure 4.17. Symbolic Coverage Legend.

The hecsym.aml program is useful for checking the output of the system. The user can zoom into an area of interest using the arcplot MAPEXTEND command. **Symcov** is then easily redisplayed by running the hecsym.aml.

5. IMPLEMENTATION OF HEC-PREPRO IN ARCVIEW.

5.1. Introduction.

The HEC-PREPRO methodology was also implemented in the ArcView environment. ArcView is multi-platform GIS software developed by the Environmental Systems Research Institute (ESRI, 1996b). ArcView provides for a programming environment via an object oriented programming language called Avenue. This chapter describes the implementation of the methodology in ArcView Avenue.

Note that in this chapter the word ‘line’ is used to describe a ‘polyline’. A ‘polyline’ is equivalent to an ‘arc’. It consists of one or more straight line segments and has a direction.

First the input data requirements are defined followed by a detailed step-by-step procedure and a description of the output data. Finally, a section on using the system serves as a user’s guide.

5.2. Input Data Description.

This section describes the input data requirements in detail. As input data a stream layer and a subbasin layer are needed. An elevation grid can be used to determine the location of the subbasin outlet for subbasins with multiple outlets. A stream location layer can be used to supply input attribute values for junction, diversion, reservoir, source and sink elements. The input data needs to be free of errors. See Section 4.2 for a detailed discussion on errors in the input data. The following sections describe the specific input data requirements for each of the four data sets.

5.2.1. Stream Layer.

- The stream layer has to be a line shape file or ARC/INFO line coverage.

- The stream layer does not have to contain node, line or polygon topology. The system itself defines the topology needed for the analysis.

- The (line) attribute table of the stream layer cannot contain fields named ‘hectype’, ‘hecid’, ‘lhp’, ‘fphp’, ‘tphp’, because they are created and used by HEC-PREPRO.

- See the discussion on the ‘tolerance distance’ in Section 5.5.5 if the stream layer has small scale errors near intersections of streams with the subbasin boundaries. This is most often the result of delineating streams and/or subbasin boundaries with grid routines.

- All lines have to point in the downstream direction. This means that the first point is upstream of the last point in the line point list. The direction of lines pointing in the upstream direction can be reversed with the ArcEdit FLIP command.

- Streams are represented by single lines. Double-line connections between two stream locations will be interpreted as reservoirs, as described below.

- Reservoirs are represented by double-line connections between two stream locations. There is only one main outlet to the reservoir

which is defined by the intersection which has two lines that are part of the reservoir pointing towards it.

5.2.2. Subbasin Layer.

- The subbasin layer has to be a polygon shape file or ARC/INFO line coverage with polygon topology.

- The subbasin layer does not have to contain node, line or polygon topology. The system itself defines the topology needed for the analysis.

- Each polygon in the layer defines a subbasin.

- Each subbasin has to contain at least part of one feature of the stream layer. In other words, each subbasin has to contain a stream.

5.2.3. Elevation Grid (Optional).

- The elevation grid has to be an ARC/INFO grid.

- The extent of the grid and the valued cells has to cover that of the stream and subbasin

5.2.4. Stream Location Layer (Optional).

- The stream location layer has to be a point shape file or ARC/INFO line coverage with node topology or ARC/INFO point coverage.

5.3. Procedure.

This section documents the procedure of the system in detail. For this discussion a type convention as shown in Table 5.1. is used.

Type	Convention	Example
Requests	*class.request* request	*list.add* request
Object	**object** (class)	**hlftab** (ftab)

Table 5.1. Type Convention.

Note that to increase readability most objects in the code are named with the class name as part of the name. The input line theme, for example, is named ‘iltheme’. Table 5.2 lists the data naming convention used for the different data sets.

Data Set	Key	Theme	Ftab	Shape Field	Shape Object
input					
Stream	il	**iltheme**	**ilftab**	**ilshapef**	**ilshape**
Subbasin	ip	**iptheme**	**ipftab**	**ipshapef**	**ipshape**
Stream Location	in	**intheme**	**inftab**	**inshapef**	**inshape**
Elevation Grid	ig	**igtheme**	n/a	n/a	n/a
output					
Hydro Line	hl	**hltheme**	**hlftab**	**hlshapef**	**hlshape**
Hydro Point	hp	**hptheme**	**hpftab**	**hpshapef**	**hpshape**
Sym Line	sl	**sltheme**	**slftab**	**slshapef**	**slshape**
Sym Point	sp	**sptheme**	**spftab**	**spshapef**	**spshape**

Table 5.2. Data Naming Convention.

The processing is done in an Avenue program called prepro.ave. A legend utility is also available. The Avenue programs are listed in Appendix B. The section numbering system used in this discussion is consistent with that in the program code. For clarity several parts of the system procedure are omitted in this discussion. They are:

- Statements to the status bar. This is done with the *application.showmsg* and *application.showstatus* requests.

- Updating of the display in the view window. This is done with the *view.draw* request.

- Updating the legend. This is done by running the **HECLEGEND** (script) program. The program is run with the *application.run* request.

- User observation level pauses. This is done with the *msgbox.info* request.

- Error checking and tracking. The **errors** (number) variable stores the number of errors encountered during program execution.

To keep the scope of the discussion manageable several repeated methods are also omitted in the discussion. They are:

- Assigning values to and determining values from attribute tables. Assignment of values is done with the *ftab.setvalue* request and determining values is done with the *ftab.returnvalue* request.

- Assigning values to and determining values from lists and dictionaries. Assignment of values is done with the *list.add* or *list.set* requests and determining values is done with the *list.get* request. A similar syntax is used for dictionaries.

- Repeated processing in the form of loops. Most loops are closely related to lists, dictionaries or tables and are therefore managed with the '*for each* **x** (object) *in* **y** (list, dictionary or ftab)' construct.

5.3.1. General Set Up.

The first step in the procedure is to do some general set up. This consists of such things as getting the input data and run control parameters, setting up the input data for further processing and reading the attribute transfer tables into memory.

5.3.1.1. Get View.

The view (**theview**, view) is obtained by sending the *application.getactivedoc* request to the **av** (application) object. Note that this assumes that the view containing the input data is active when the script is run.

5.3.1.2. Get Themes.

A list of active themes (**theactivethemes**, list) is obtained by sending the *view.getactivethemes* to **theview** (view). Note that this assumes that the input themes are active in the view. Then **theactivethemes** (list) is searched for the individual input themes. See Section 5.2 for a detailed description of the input data. Table 5.3 lists the input themes and criteria for finding them.

Input Theme	Object Name	Found Variable	Criteria
Stream	**iltheme**	**ilfound**	line feature theme (ftheme)
Subbasin	**iptheme**	**ipfound**	polygon feature theme (ftheme)
Stream Location*	**intheme**	**infound**	point feature theme (ftheme)
Elevation Grid*	**igtheme**	**igfound**	grid theme (gtheme)

*optional input data.

Table 5.3. Input Theme Determination.

5.3.1.3. Get Run Control Parameters.

The user is prompted to supply the run control parameters with the *msgbox.multiinput* request. Table 5.4 lists the run control parameters. See Section 5.5.3 for a detailed description of each of the parameters.

Variable	Class	Assignment from Input	Default
attrib	boolean	if **input**.*left(1).ucase* = "N" false else true	no
hmsmode	string	if **input**.*left(3).ucase* = "DEF" 'd' else **input**	default
tol	number	any number in map units	10
oblevel	number	any number	2

Table 5.4. Run Control Parameters.

5.3.1.4. Set Up Hydrologic Element Type Dictionary.

The hydrologic element type dictionary (**hedict**, dictionary) lists the numbers and names of the hydrogic element types. This dictionary will be used in subsequent calculations. Table 5.5 lists the dictionary structure and Table 5.6 lists the values.

Position	Value	Class
key	hydrologic element type	number
0	hydrologic element type	string

Table 5.5. Hydrologic Element Type Dictionary Structure.

Key	Value
5	Subbasin
4	Reach
3	Junction
8	Diversion
10	Reservoir
7	Source
6	Sink

Table 5.6. Hydrologic Element Type Dictionary Values.

5.3.1.5. Set Up Input Themes.

Setting up the input themes consists of getting the attribute table of each feature theme by sending it the *ftheme.getftab* request. The shape and attribute fields are obtained by sending the *ftab.getfields* request to each of the attribute tables. See Table 5.2 in beginning of this section for the object names.

5.3.1.6. Get Attribute Map Info.

This is only done if attributes are transferred as specified in the **attrib** (boolean) variable. The attribute transfer information is stored in the attribute

transfer tables. Each hydrologic element type has its own attribute transfer table. See Section 5.5.4 for a detailed description of the attribute transfer methodology.

The attribute transfer or mapping information is read into attribute transfer dictionary (**mapdict**, dictionary). Table 5.7 lists the **mapdict** (dictionary) structure.

Position	**Value**	**Class**
key	hydrologic element type	string
0	map attributes?	Boolean
1	table name	string
2	table vtab	vtab
3	field list (see Table 5.8 for structure)	list

Table 5.7. Attribute Transfer Dictionary Structure.

Position	Value	Class
0	Name of field in HEC-HMS Basin file.	String
1	Name of field in input data	string
2	Transfer mode (see Table 5.27 in Section 5.5.4 for values)	number

Table 5.8. Attribute Transfer Field List Structure.

First a list of all the tables in the project (**thetables**, list) is obtained by checking the class of all the documents (**thedocs**, doc) in the project. This is done with the *project.getdocs*, *doc.getclass* and *class.getclassname* requests. Then the name of each table is checked against the list of names (Table 5.28, Section 5.5.4) using the *object.getname* request. Then each table is processed and **mapdict** (dictionary) is populated.

5.3.2. Create Hydro Line Shape File.

The stream line layer (**iltheme**) is intersected with the subbasin polygon layer (**iptheme**). Note that the procedure in this section is based on a script called 'View.IntersectThemes' written by ESRI. The script supports feature

selection and projections. Support of feature selection means that if the input theme has several selected features the code will only operate on those features. Support of projections means that the code will operate on the features in the projection defined in the view. These properties of the code have been retained in this part of the system. The remainder of the system does however not support them.

5.3.2.1. Set Up Hydro Line Theme.

The result of the intersection is the hydro line theme (**hltheme**, ftheme). The file name for the hydro line theme (**hlfilename**, filename) is assigned with the *project.makefilename* request, which checks if the name allready exists and adds a number to the name if it does exist. If, for example, a shape file named 'hydrol7.shp' exists the *project.makefilename* creates a filename called 'hydrol8.shp'. Note that this way data from previous runs will not be overwritten. The hydro line attribute table (**hlftab**, ftab) is created with the *ftab.makenew* request. Working fields are created using the *field.make* request and added to **hlftab** (ftab) using the *ftab.addfields* request. Table 5.9 lists the working fields.

Field Name	Field Type	Field Length/ Precision	Description
hectype	number	16/4	hydrologic element type
hecid	number	16/4	ID
lhp	number	16/4	record number (**hpftab**) of reservoir outlet point
fphp	number	16/4	record number (**hpftab**) of point located at upstream end of stream
tphp	number	16/4	record number (**hpftab**) of point located at downstream end of stream

Table 5.9. Hydro Line Theme Working Fields.

If attributes are transfered (**attrib**, boolean) the fields of the input stream line attribute table (**ilftab**, ftab) are also added to **hlftab** (ftab). The 'shape' field (**hlshapef**, field) in the attribute table is obtained with the *ftab.findfield* request. Finally the hydro line attribute table (**hlftab**, ftab) is made editable using the *ftab.seteditable* request.

5.3.2.2. Intersect.

First the selected features in the stream and subbasin themes are assigned to working objects (**theme1**, **theme2**, fthemes). This done to support feature selection. The code only operates on selected features.

Each polygon in the temporary subbasin theme (**theme2**, ftheme) is processed. Note the syntax to support projections. All the lines in the temporary stream theme (**theme1**, ftheme) that are fully or partially within the polygon are selected using the *ftab.selectbyshapes* request.

Then each of the selected lines in the temporary stream theme (**theme1**, ftheme) is checked to see if it is fully contained inside the polygon using the *shape.iscontainedin* request. If the line is fully contained in the polygon it is added to the hydro line theme (**hltheme**, ftheme). If it is not fully contained in the polygon the part of the line that is inside the polygon is clipped of the line using the *shape.lineintersection* request and that line is then added to the hydro line theme (**hltheme**, ftheme). Adding a line is done by creating a new record with the *ftab.addrecord* request and writing the line shape to the 'shape' field using the *ftab.setvalue* request. The attribute values

of the line in the stream line theme (**iltheme**, ftheme) are written to the attribute fields of the line in the hydro line theme (**hltheme**, ftheme).

5.3.3. Create Hydro Point Shape File.

The hydro point theme (**hptheme**, ftheme) is a point shape file with points at the intersection of lines in the hydro line theme (**hltheme**, ftheme). In other words the points in the hydro point theme (**hptheme**, ftheme) are nodes for the hydro line theme (**hltheme**, ftheme).

5.3.3.1. Set Up Hydro Point Theme.

Similar to the set up for the hydro line theme (**hltheme**, ftheme) this includes the specification of the file name, creation of ftab and addition of working fields. Working fields are listed in Table 5.10.

Field Name	Field Type	Field Length/ Precision	Description
hecid	number	16/4	ID
hectype	number	16/4	hydrologic element type
rnt	number	16/4	node type taking into account all stream lines
hnt	number	16/4	node type taking into acount channel system stream lines
rl	number	16/4	lines connecting
rupl	number	16/4	lines connecting ustream
rdnl	number	16/4	lines connecting downstream
rrl	number	16/4	lines from reservoir connecting
rrupl	number	16/4	lines from reservoir connecting upstream
rrdnl	number	16/4	lines from reservoir connecting downstream
hl	number	16/4	lines from channel system connecting
hupl	number	16/4	lines from channel system connecting upstream
hdnl	number	16/4	lines from channel system connecing downstream
hrl	number	16/4	lines from channel system and reservoir connecting
hrupl	number	16/4	lines from channel system and reservoir connecting upstream
hrdnl	number	16/4	lines from channel system and reservoir connecting downstream

Table 5.10. Hydro Point Theme Working Fields.

Attribute fields from the stream location attribute table (**inftab**, ftab) are added to the hydro point attribute table (**hpftab**, ftab) if such a theme was provided as input data (**infound**, boolean). Finally the hydro point attribute table (**hpftab**, ftab) is made editable using the *ftab.seteditable* request.

5.3.3.2. Add Points.

Each line in the hydro line theme (**hltheme**, ftheme) is processed. The shape of the line is converted to a listing of points using the *polyline.asmultipoint* and *multipoint.aslist* requests. For the beginning and ending points of the line (**hlfp**, **hltp**, points) the existing points in the hydro point theme (**hptheme**, ftheme) are checked. If no point exists within the tolerance (**tol**, number) a point is added at that location. Similar to adding a line adding a point is done with the *ftab.addrecord* and *ftab.setvalue* requests. The record number of the new point is written to the 'fphp' field or 'tphp' field of the line in the hydro line theme (**hltheme**, ftheme).

If a stream location theme (**intheme**, ftheme) was specified as input data (**infound**, boolean) the points in that theme are also checked. If a point exists

within the tolerance distance (**tol**, number) of the new point the attribute values of the point in the stream location theme (**intheme**, ftheme) are written to the attribute fields of the point in the hydro point theme (**hptheme**, ftheme).

5.3.4. Identify Sources, Subbasin Outlets and Sinks.

Points located on the boundary line of a subbasin represent the location of source, subbasin outlet or sink elements. Each point in the hydro point theme (**hptheme**, ftheme) is processed. For each point all the polygons from the subbasin theme (**iptheme**, ftheme) are checked. The polygon shape (**polyshape**, polygon) is converted to a (poly)line shape (**polylshape**, polyline) with the *polygon.aspolyline* request. The distance from the point (**hpshape**, point) to the (poly)line (**polylshape**, polyline) is checked. If it is less than the tolerance (**tol**, number) then the 'hectype' field in the hydro point attribute table (**hpftab**, ftab) is set to '1'.

5.3.5. Identify Channel System.

The channel system consists of the stream lines downstream of hydrologic elements. This is obtained by tracing downstream of the source, subbasin outlet and sink locations established in the previous step.

5.3.5.1. Create Stream Dictionary.

The stream dictionary (**streamdict**, dictionary) consists of all the stream lines that are longer than half of the tolerance (**tol**, tolerance). See Section 5.5.5 for detailed discussion on the tolerance. The dictionary is structured as shown in Table 5.11.

Position	Value	Object Type
key	record number (**hlftab**, ftab) of stream line	number
0	record number (**hlftab**, ftab) of stream line	number

Table 5.11. Stream Dictionary Structure.

To create the stream dictionary (**streamdict**, dictionary) each record in the hydro line theme (**hltheme**, ftheme) is processed. The length of the line (**hllength**, number) is calculated from the list of points making up the line. If length is longer than half of the tolerance (**tol**, number) the record number of the line is added to the stream dictionary (**streamdict**, dictionary).

5.3.5.2. Create Channel Dictionary.

The channel dictionary (**channeldict**, dictionary) consists of all the stream lines in the stream dictionary (**streamdict**, dictionary) that are part of the channel system. It has a structure identical to **streamdict** (dictionary) as shown in Table 5.11.

Each point in the hydro point theme (**hptheme**, ftheme) is processed. If the point is a source, subbasin outlet or sink (hectype = 1) the analysis proceeds otherwise the next point is processed.

Each line in the stream dictionary (**streamdict**, dictionary) is processed. If the distance from the beginning point of the line to the point is less than the tolerance (**tol**, number) the line is downstream of a source, subbasin outlet or sink point and therefore part of the channel system. The **hlftab** (ftab) record number of the line is added to the channel dictionary (**channeldict**, dictionary) and the upstream line list (**uplist**, list) which will be used for tracing (described in the next paragraph). Then the **hlftab** (ftab) record number of all the lines in the channel dictionary (**channeldict**, dictionary) are removed from the stream dictionary (**streamdict**, dictionary). This is done to decrease the number of lines processed for the next point. Figure 5.1 lists the relevant Avenue code.

```
for each hprec in hpftab
    if (hpftab.returnvalue(hphectypef, hprec) = 1) then
        hpshape = hpftab.returnvalue(hpshapef, hprec)
        for each hlrec in streamdict
            hlshape = hlftab.returnvalue(hlshapef, hlrec)
            hlpoints = hlshape.asmultipoint.aslist
            if (hlpoints.get(0).distance(hpshape) < tol) then
                channeldict.add(hlrec.clone, hlrec.clone)
                uplist = uplist.add(hlrec.clone)
            end
        end
        for each hlrec in channeldict
            streamdict.remove(hlrec)
        end
    end
end
```

Figure 5.1. Channel System Upstream Line Identification Avenue Code.

The previous paragraph described how the record number of all the lines downstream of source, subbasin outlet or sink elements are written to the channel dictionary (**channeldict**, dictionary) and the upstream line list (**uplist**, list). To identify the complete channel system those lines are traced downstream. The tracing procedure is as follows.

Each line in the upstream line list (**uplist**, list) is processed. The ending point of the line is compared to the beginning point of all the lines in the stream dictionary (**streamdict**, dictionary). If the distance between them is less than the tolerance (**tol**, number) the line in **streamdict** (dictionary) is downstream of the line in **uplist** (list) and its **hlftab** (ftab) record number is

added to the downstream line list (**dnlist**, list) as well as the channel dictionary (**channeldict**, dictionary). Then all the lines from the channel dictionary (**channeldict**, dictionary) are removed from the stream dictionary (**streamdict**, dictionary) to reduce processing time for the next line. If no downstream lines are found the process is ended, otherwise the downstream line list (**dnlist**, list) becomes the new upstream line list (**uplist**, list) and the procedure is repeated. Figure 5.2 shows the relevant Avenue code.

```
done = false
while (not done)
    dnlist = list.make
    for each uplrec in uplist
        uplineshape = hlftab.returnvalue(hlshapef, uplrec)
        uplinepoints = uplineshape.asmultipoint.aslist
        updnpoint = uplinepoints.get(uplinepoints.count - 1)
        for each hlrec in streamdict
           hlshape = hlftab.returnvalue(hlshapef, hlrec)
           if (hlshape.asmultipoint.aslist.get(0) ...
                     ... .distance(updnpoint) < tol) then
               dnlist = dnlist.add(hlrec.clone)
               channeldict.add(hlrec.clone, hlrec.clone)
           end
        end
        for each hlrec in channeldict
            streamdict.remove(hlrec)
        end
    end
    if (dnlist.count = 0) then
        done = true
    end
    uplist = dnlist
end
```

Figure 5.2. Channel System Tracing Avenue Code.

5.3.5.3. Mark Lines.

Each line in the channel dictionary (**channeldict** dictionary) is processed. The corresponding lines in the hydro line theme (**hltheme**, ftheme) are assigned hydrologic element type numbers for reaches (hectype = 4).

5.3.6. Identify Lakes.

Lakes are defined by polygons in the stream line layer (**iltheme**, ftheme). Since this program does not use any pre-existing topology and topology building functions are not directly available in ArcView the lakes are identified from the lines in the hydro line theme (**hltheme**, ftheme).

In short, the procedure is to look for a diversions in the stream lines. If a diversion is found, each downstream path is traced. If the two paths meet somewhere downstream in a junction a lake is present. The lines constituting

the lake are identified by tracing each upstream path from the junction to the diversion. Figure 5.3 illustrates the methodology.

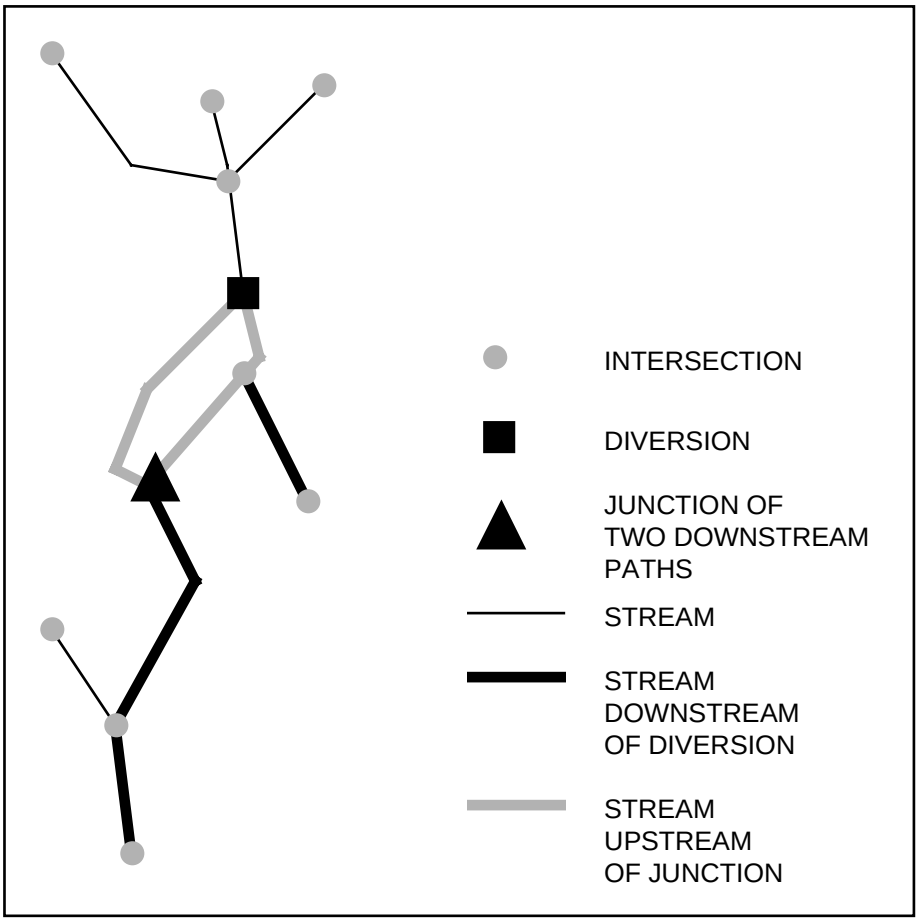

Figure 5.3. Lake Identification Methodology.

Each point in the hydro point theme (**hptheme**, ftheme) is processed. For each point all the lines in the hydro line theme (**hltheme**, ftheme) are processed (if they are longer than half the tolerance distance, see Section 5.5.5). The **hpftab** (ftab) record numbers of the points located at the beginning and ending points were written to the 'fphp' and 'tphp' fields in the hydro line attribute table (**hlftab**, ftab) in the previous step. Those values are checked against the record number of the point. If the line is upstream or downstream of the point a corresponding counter (**uplines**, **dnlines**, numbers) is increased by 1. At the end of the loop the counters indicate the number of lines upstream and downstream of the point.

If the number of lines upstream and downstream of the point are one and two respectively the point is a diversion and possibly and upstream end of a lake. If that is the case the **hlftab** (ftab) record numbers of the two downstream lines are added to a trace list (**tracelist**, list) which is used to store the **hlftab** (ftab) record numbers of the lines that are being traced. If it later turns out we are dealing with a polygon this list will contain all the **hlftab** (ftab) record numbers of the lines belonging to the polygon.

The tracing occurs simultaneously along both (possible) shores of the lake as well as further diversions encountered along the way. The **hlftab** (ftab) record number of the current lines are stored in the upstream line list (**uplist**, list). Then each line in the hydro line theme (**hltheme**, ftheme) is processed. If the hydro point number of its upstream end matches the hydro point number of the downstream end of the line from **uplist** (list) it is added to a downstream line list (**dnlist**, list). Once all the lines have been processed **dnlist** (list) becomes the new **uplist** (list) and the process continues until no more downstream lines are found. During this process all the lines traced are also added to **tracelist** (list). Figure 5.4 shows the essential downstream lake tracing Avenue code.

```
tracelist = list.make
tracelist.add(first downstream line of diversion)
tracelist.add(second downstream line of diversion)
uplist = list of downstream lines of diversion
done = false
while (not done)
    dnlist = list.make
    for each uplrec in uplist
        upltphp = hlftab.returnvalue(hltphpf, uplrec)
        for each hlrec in hlftab
            hlfphp = hlftab.returnvalue(hlfphpf, hlrec)
            if (hlfphp = upltphp) then
                dnlist = dnlist.add(hlrec.clone)
                tracelist.add(hlrec.clone)
            end
        end
    end
    if (dnlist.count = 0) then
       done = true
    end
    uplist = dnlist
end
```

Figure 5.4. Lake Downstream Tracing Avenue Code.

Then a copy of the tracelist (**tracelistb**, list) is made and the *list.removeduplicates* request is used to delete all the duplicates in the list. All the records in **tracelist** (list) are compared to the records in **tracelistb** (list). If the records don't match a line was traversed twice indicating a junction in the system. If that occurs it is established that we are dealing with a lake. The **hlftab** (ftab) record number of the line found to be processed twice is remembered as **dnline** (number).

If a lake is found as described above all the records in **tracelist** (list) are traced upstream from the junction point, which is the upstream point of **dnline** (number). The tracing procedure is in essence the same as the downstream tracing procedure from above. The lines that are part of the lake are stored in **tracelistc** (list).

All the records in **tracelistc** (list) are processed. The hectype of the line is multiplied by a factor of 10. Lines that are part of the reservoir therefore have a hectype of 20 or 40 depending on if they are or are not part of the channel system. Further the **hpftab** (ftab) record number of the downstream point of the lake is written to the 'lhp' field of each of the lines.

5.3.7. Identify Channel Elements.

5.3.7.1. Set Up Sym Dictionaries.

The sym dictionaries are created for storing the watershed data in the hydrologic data structure before they are written to the sym line and point

themes and the HEC-HMS Basin file. The structure of the dictionaries is described in Tables 5.12 through 5.15.

Position	Value	Object Type
key	hecid	number
0	hecid	number
1	hectype	number
2	hecupid	number
3	hecdnid	number
4	hecupx	number
5	hecupy	number
6	hecdnx	number
7	hecdny	number

Table 5.12. Sym Line Dictionary Structure.

Position	Value	Object Type
key	hecid	number
0	hecid	number
1	hectype	number
2	hecupids	number
3	hecdnids	number
4	hecx	number
5	hecy	number
6	hprec	number

Table 5.13. Sym Point Dictionary Structure.

Position	Value	Object Type
key	hecid	number
0	hecid	number
1	hecupid 1	number
2	hecupid 2	number
3	etc...	number

Table 5.14. Upstream ID Dictionary Structure.

Position	Value	Object Type
key	hecid	number
0	hecid	number
1	hecdnid 1	number
2	hecdnid 2	number
3	etc...	number

Table 5.15. Downstream ID Dictionary Structure.

5.3.7.2. Set Up Hecid Variable.

The **hecid** (number) variable is initiated by setting it equal to ‘1’.

5.3.7.3. Identify Elements.

Each point in the hydro point theme (**hptheme**, ftheme) is processed. For each point each line in the hydro line theme (**hltheme**, ftheme) is processed. The length of the line (**hllength**, number) is calculated. If the length of the line (**hllength**, number) is larger than half the tolerance (**tol**, number) the spatial relation of the line to the point is examined. A connection between the beginning or ending point of the line and the point occurs if the distance among them is less than the tolerance distance. The type of the line (channel system, reservoir, etc.) is checked and corresponding counters are incremented. Counters and their matching fields in the hydro point theme (**hptheme**, ftheme) are listed in Table 5.16. Further description of the counters can be found in Table 5.10.

Variable	Hydro Point Theme Field
rlines	rl
ruplines	rupl
rdnlines	rdnl
rrlines	rrl
rrupllines	rrupl
rrdnlines	rrdnl
hlines	hl
hdnlines	hdnl
hrlines	hrl
hruplines	hrupl
hrdnlines	hrdnl

Table 5.16. Hydrotopology Counters.

The node types are calculated and written to the hydro point attribute table (**hpftab**, ftab) along with the values of the counters. See Section 3.4.4 for further description of the node types.

The point is checked against the identification criteria of the hydrologic element types. If the point is a hydrologic element the hecid and corresponding hectype number is written to the hydro point attribute table (**hpftab**, ftab) and the sym point dictionary (**spdict**, dictionary). The coordinates of the point are also written to the sym point dictionary (**spdict**, dictionary). A record is added to the upstream and downstream ID

dictionaries (**upiddict**, **dniddict**, dictionaries). Lastly, the hecid counter (**hecid**, number) is incremented by '1'.

5.3.8. Establish Connectivity.

5.3.8.1. Reaches and Channel Elements.

In general, connectivity of reaches and channel elements is established by picking a line in the channel dictionary (**channeldict**, dictionary) that does not belong to a lake and tracing upstream and downstream until a channel element or a reservoir is found. All the lines traced in this way are combined into a single reach by assigning the same **hecid** (number) to them.

First a dictionary of lines to be processed (**processdict**, dictionary) is created as a copy of the channel dictionary (**channeldict**, dictionary). Then a dictionary of all the possible starting points (**startdict**, dictionary) for the procedure is created by copying **channeldict** (dictionary) and removing all the lines belonging to a lake. Lines part of a lake are identified by having a non-zero value in the 'lhp' field in the hydro line attribute table (**hlftab**, ftab).

As long as there are records in **startdict** (dictionary) the following procedure is performed. A reach list (**reachlist**, list) is created. The first record from **startdict** (dictionary) is added to **reachlist** (list). Then the channel system is traced upstream. The tracing procedure is similar to the one used for the lakes.

A downstream line number (**dnline**, number) is created and assigned the value of the line from **startdict** (dictionary). The hectype of the upstream point of the line is checked against possible reach upstream end hectype values. The upstream point of a reach can have a hectype of 1 (subbasin outlet), 3 (junction), 7 (source), 8 (diversion) or 10 (reservoir). If the upstream point of the line does not meet the criteria for upstream end of a reach the following tracing procedure is done until an upstream point of a reach or a reservoir is found.

Each line in **processdict** (dictionary) is processed. The **hpftab** (ftab) record number of the ending point of the line is compared to the **hpftab** (ftab) record number of the starting point of the **dnline** (number). If the numbers match and the line is not part of a reservoir the line is added to **reachlist** (list).

The hectype of the upstream end of the line is checked to see if it is possible upstream end of a reach (hectype 1, 3, 7, 8 or 10). If the upstream end of the line is a possible upstream end of a reach the loop is ended, otherwise continued.

For non-lake upstream ends the hecid value as well as the shape and coordinates of the upstream end are written to memory. For lake upstream ends the reservoir point is found from the 'lhp' value of the line. The lines in **reachlist** (list) are removed from **processdict** (dictionary) and **startdict** (dictionary). Figure 5.5 illustrates the relevant Avenue code.

```
dnline = reachlist.get(0).clone
upfound = false
uplake = false
hlfphp = hlftab.returnvalue(hlfphpf, dnline)
uphectype = hpftab.returnvalue(hphectypef, hlfphp)
if (uphectype = 1) then
    upfound = true
end
if (uphectype = 3) then
    upfound = true
end
if (uphectype = 7) then
    upfound = true
end
if (uphectype = 8) then
    upfound = true
end
if (uphectype = 10) then
    upfound = true
end
while ((not upfound) and (not uplake))
    for each prec in processdict
        hltphp = hlftab.returnvalue(hltphpf, prec)
        if (hltphp = hlfphp) then
            hllhp = hlftab.returnvalue(hllhpf, prec)
            if (hllhp = 0) then
                reachlist.add(prec.clone)
            end
            if (hllhp > 0) then
                uplake = true
            end
            dnline = prec.clone
            break
        end
    end

...
```

Figure 5.5. Reach Tracing Avenue Code (continued on next page).

```
...
    hlfphp = hlftab.returnvalue(hlfphpf, dnline)
    uphectype = hpftab.returnvalue(hphectypef, hlfphp)
    if (uphectype = 1) then
        upfound = true
    end
    if (uphectype = 3) then
        upfound = true
    end
    if (uphectype = 7) then
        upfound = true
    end
    if (uphectype = 8) then
        upfound = true
    end
    if (uphectype = 10) then
        upfound = true
    end
end
if (not uplake) then
    hecupid = hpftab.returnvalue(hphecidf, hlfphp)
    hecupshape = hpftab.returnvalue(hpshapef, hlfphp)
    hecupx = hecupshape.getx
    hecupy = hecupshape.gety
end
if (uplake) then
    hecupid = hpftab.returnvalue(hphecidf, hllhp)
    hecupshape = hpftab.returnvalue(hpshapef, hllhp)
    hecupx = hecupshape.getx
    hecupy = hecupshape.gety
end
for each hlrec in reachlist
    processdict.remove(hlrec)
    startdict.remove(hlrec)
end
```

Figure 5.5. Reach Tracing Avenue Code (continued from previous page).

The same procedure is followed in the downstream direction. Then the lines in **reachlist** (list) are assigned identical hecid values in the hydro line attribute table (**hlftab**, ftab). The **hecid**, **hecupid**, **hecdnid**, **hecupx**, **hecupy**,

hecdnx and **hecdny** (numbers) values are written to the sym line dictionary (**symdict**, dictionary). The **spdict** (dictionary) records of the upstream and downstream elements are updated. The **hecid** (number) is incremented by '1'. The number of lines in **startdict** (dictionary) is checked with the *dictionary.count* request. If there are no more possible starting lines the loop ends.

5.3.8.2. Subbasins.

The subbasin elements have to be connected to the corresponding subbasin outlet elements. Each subbasin in the subbasin theme (**iptheme**, ftheme) is processed. For each subbasin each point in the hydro point theme (**hptheme**, ftheme) is processed. If the distance from the point to the boundary line of the subbasin is less than the tolerance (**tol**, number) the point is located on the subbasin boundary. If that is the case the analysis proceeds otherwise the next point is processed.

Each line in the hydro line theme (**hltheme**, ftheme) is processed. If the ending point of the line is within the tolerance (**tol**, number) of the point and

the line is completely contained in the polygon the point is an outlet of the subbasin. The *shape.iscontainedin* request is used to check if the line is completely contained in the polygon. The **hpftab** (ftab) record number of the point is added to the outlet point list (**outletlist**, list). If there is only one record in the outlet point list (**outletlist**, list) that point is chosen to be the subbasin outlet. If there are multiple outlet points to the subbasin the first point is chosen to be the subbasin outlet, unless an elevation grid was specified as input data (**igfound**, boolean). If an elevation grid was supplied the subbasin outlet is determined as described in the next paragraph.

For subbasins with multiple outlets a choice has to be made as to which outlet is the subbasin outlet. The elevation of the point is used as criteria. The outlet with the lowest elevation is chosen to be the subbasin outlet. The elevation of the point is extracted from the elevation grid (**igtheme**, gtheme) with the *gtheme.returncellvalue* request.

5.3.9. Create Sym Line Shape File.

5.3.9.1. Set Up Sym Line Theme.

The file name for the sym line theme (**sltheme**, ftheme) is created with the *project.makefilename* request, and the ftab is created using the *ftab.makenew* request. The shape field (**slshapef**, field) is obtained with the *ftab.findfield* request. A number of working fields are added to the sym line attribute table (**slftab**, ftab) as listed in Table 5.17.

Field Name	Field Type	Field Length/ Precision	Description
hecid	number	16/4	ID
hectype	number	16/4	hydrologic element type
hecupid	number	16/4	ID of upstream element
hecdnid	number	16/4	ID of downstream element
hecupx	number	16/4	x coordinate of upstream element
hecupy	number	16/4	y coordinate of upstream element
hecdnx	number	16/4	x coordinate of downstream element
hecdny	number	16/4	y coordinate of downstream element

Table 5.17. Sym Line Theme Working Fields.

If attributes are transferred as indicated by **attrib** (boolean) the attribute fields specified in the reach attribute transfer table are also added to the sym line attribute table (**slftab**, ftab). The field type, length and precision is copied from the stream line attribute table (**ilftab**, ftab).

5.3.9.2. Add Lines.

Lines are added to the sym line theme (**sltheme**, ftheme) based on the sym line dictionary (**sldict**, dictionary). The *ftab.addrecord* and *ftab.setvalue* requests are used to add a line. Attribute values from the lines in the stream line attribute table (**ilftab**, ftab) are transferred to the attribute fields in the sym line attribute table (**slftab**, ftab) based on the reach attribute transfer table. See the discussion in Section 5.5.4 for a detailed description of the reach attribute transfer methodology.

5.3.10. Create Sym Point Shape File.

5.3.10.1. Set Up Sym Point Theme.

The file name for the sym point theme (**sptheme**, ftheme) is created with the *project.makefilename* request, and the ftab is created using the *ftab.makenew* request. The shape field (**spshapef**, field) is obtained with the *ftab.findfield* request. A number of working fields are added to the sym point attribute table (**spftab**, ftab) as listed in Table 5.18.

Field Name	Field Type	Field Length/ Precision	Description
hecid	number	16/4	ID
hectype	number	16/4	hydrologic element type
hecupids	number	16/4	number of upstream elements
hecdnids	number	16/4	number of downstream elements
hecx	number	16/4	x coordinate
hecy	number	16/4	y coordinate

Table 5.18. Sym Point Theme Working Fields.

If attributes are transferred as indicated by **attrib** (boolean) the attribute fields specified in the subbasin, junction, diversion, reservoir, source and sink

attribute transfer table are also added to the sym line attribute table (**slftab**, ftab). The field type, length and precision is copied from the input attribute tables.

5.3.10.2. Add Points.

Points are added to the sym point theme (**sptheme**, ftheme) based on the sym point dictionary (**spdict**, dictionary). The *ftab.addrecord* and *ftab.setvalue* requests are used to add a point. Attribute values from the polygons in the subbasin polygon attribute table (**ipftab**, ftab) and the stream location point attribute table (**inftab**, ftab), if provided, are transferred to the attribute fields in the sym point attribute table (**spftab**, ftab) based on the element attribute transfer table. See the discussion in Section 5.5.4 for a detailed description of the reach attribute transfer methodology.

5.3.11. Link Tables.

The attribute tables of the sym line and point themes (**sltheme**, **sptheme**, fthemes) are linked to the attribute tables of the hydro line and point themes (**hltheme**, **hptheme**, fthemes), respectively. The link is based on 'hecid' and established using the *ftab.link* request.

5.3.12. Create HEC-HMS Basin File.

Set Up File: If the **hmsmode** (string) run control variable is set to 'd', the filename for the HEC-HMS Basin file (**hmsfilename**, filename) is created with the *project.makefilename* request, otherwise the value of **hmsmode** (string) is used to create the **hmsfilename** (filename) with the *string.asfilename* request. The HEC-HMS Basin file (**hmsfile**, file) is created using the *linefile.make* request.

Write Header: The file header is written. The date and time is obtained from the system using the *date.now* request. The units system is set

as ‘unknown’. Data are written to the file one line at a time using the *file.writeelt* request

Write Element Data: Element data are written to the HEC-HMS Basin file based on the sym line and point themes (**sltheme**, **sptheme**, fthemes). See the discussion in Section 5.5.4 for a detailed description of the attribute transfer methodology.

Close File: The HEC-HMS Basin file (**hmsfile**, file) is closed by sending it the *file.close* request.

5.3.13. Close Up.

The attribute tables of the hydro and sym line and point themes (**hlftab**, **hpftab**, **slftab**, **spftab**, ftabs) are closed or made non-editable using the *ftab.seteditable* request.

5.4. Output Data Description.

This section describes the output data in detail. The output data consists of an HEC-HMS basin file, and a 'hydrologic' and a symbolic shape file.

5.4.1. HEC-HMS Basin File Output.

See Section 3.4.7 for a description of the HEC-HMS Basin File. As a minimum the HEC-HMS Basin file contains information to define the hydrologic elements, their location and connectivity among them. Table 5.19 lists the fields always contained in the HEC-HMS Basin file. The file also contains attributes as specified in the attribute transfer tables. See Section 5.5.4 for a description of the attribute transfer capabilities.

Hydrologic Element Type	Minimum Fields Valued in HEC-HMS Basin File.
Subbasin	Subbasin
	Canvas X
	Canvas Y
	Downstream
Reach	Reach
	Canvas X
	Canvas Y
	From Canvas X
	From Canvas Y
	Downstream
Junction	Junction
	Canvas X
	Canvas Y
	Downstream
Diversion	Diversion
	Canvas X
	Canvas Y
	Downstream
Reservoir	Reservoir
	Canvas X
	Canvas Y
	Downstream
Source	Source
	Canvas X
	Canvas Y
	Downstream
Sink	Sink
	Canvas X
	Canvas Y

Table 5.19. Minimum Data Written to HEC-HMS Basin File.

5.4.2. Hydrologic Line Shape File Output.

A hydrologic (hydro) line shape file is created. The file is the result of the intersection of the stream line layer and the subbasin polygon layer. Several working fields are added to the attribute table. The attribute fields from the stream line layer are also in the hydro line attribute table. The attribute values from the stream lines are written to the corresponding attribute fields of the hydro lines. Table 5.20 lists the attribute fields.

Field Name	**Description**
hectype	hydrologic element type
hecid	ID
lhp	hydro point record number of reservoir outlet point
fphp	hydro point record number of point located at upstream end of stream
tphp	hydro point record number of point located at downstream end of stream
input	*all fields from stream line layer*

Table 5.20. Hydro Line Theme Fields.

5.4.3. Hydrologic Point Shape File Output.

A hydrologic (hydro) point shape file is created. The file has points located at the intersection of lines of the hydro line shape file. In ARC/INFO terminology the points in the hydro point shape file are nodes of the lines in the hydro line shape file. Several working fields are added to the attribute table. Also, if a stream location theme was selected as input data the attribute values of any point in the stream location theme that lies within the tolerance of a point in the hydro point theme are written to the attribute fields of the point in the hydro point theme. See Section 5.5.4 for a detailed discussion of the attribute transfer methodology. Table 5.21 lists the attribute fields.

Field Name	Description
hecid	ID
hectype	hydrologic element type
rnt	node type taking into account all stream lines
hnt	node type taking into acount channel system stream lines
rl	lines connecting
rupl	lines connecting ustream
rdnl	lines connecting downstream
rrl	lines from reservoir connecting
rrupl	lines from reservoir connecting upstream
rrdnl	lines from reservoir connecting downstream
hl	lines from channel system connecting
hupl	lines from channel system connecting upstream
hdnl	lines from channel system connecing downstream
hrl	lines from channel system and reservoir connecting
hrupl	lines from channel system and reservoir connecting upstream
hrdnl	lines from channel system and reservoir connecting downstream
input	*all fields from stream location layer*

Table 5.21. Hydro Point Theme Fields.

5.4.4. Symbolic Line Shape File Output.

A symbolic (sym) line shape file is created. The lines represent reach elements or links. Several working items are added to the attribute table of the shape file. Also attribute fields as specified in the attribute transfer tables are added to the attribute table of the shape file. The attribute values from the corresponding stream lines in the hydro line shape file are transferred to the attribute fields of the lines in the sym line shape file. There are several options available for attribute transfer as described in Section 5.5.4. Table 5.22 lists the attribute fields.

Field Name	Description
hecid	ID
hectype	hydrologic element type
hecupid	ID of upstream element
hecdnid	ID of downstream element
hecupx	x coordinate of upstream element
hecupy	y coordinate of upstream element
hecdnx	x coordinate of downstream element
hecdny	y coordinate of downstream element
input	*all items specified in the reach attribute transfer table*

Table 5.22. Sym Line Theme Fields.

5.4.5. Symbolic Point Shape File Output.

A symbolic (sym) point shape file is created. The lines represent subbasin, junction, diversion, reservoir, source or sink elements. Several working items are added to the attribute table of the shape file. Also attribute fields as specified in the attribute transfer tables are added to the attribute table of the shape file. The attribute values from the corresponding subbasin polygon, if

provided as input data, from the corresponding stream location point are transferred to the attribute fields of the points in the sym point shape file. See Section 5.5.4 for a detailed discussion of the attribute transfer methodology. Table 5.23 lists the attribute fields.

Field Name	Description
hecid	ID
hectype	hydrologic element type
hecupids	number of upstream elements
hecdnids	number of downstream elements
hecx	x coordinate
hecy	y coordinate
input	*all fields specified in the subbasin, junction, diversion, reservoir, source and sink attribute transfer tables*

Table 5.23. Sym Point Theme Fields.

5.5. Using the HEC-PREPRO System.

5.5.1. Getting the System.

The system consists of Avenue programs, also called scripts, that run inside ArcView. The Spatial Analyst is optional. If the Spatial Analyst is installed a Grid can be specified as input elevation grid for additional capabilities, as discussed later in this section. The system consists of the following two Avenue programs:

- Main Program (hecprepro.ave).
- Legend Utility (heclegend.ave).

The programs can be downloaded via anonymous ftp as described in Table 5.24.

Data	Value
Site	ftp.crwr.utexas.edu
Login	anonymous
Password	*your e-mail address*
Directory	/pub/crwr/gishydro/hecprepro/
Mode	ASCII
Files	hecprepr.ave and heclegen.ave *or* prepro.apr

Table 5.24. System Download Information.

5.5.2. Installing the System.

To install the system the scripts have to be loaded into script editors and compiled. The hecprepro.ave and heclegend.ave scripts have to be named 'HECPREPRO' and 'HECLEGEND', respectively. The scripts can be tied to buttons on the view button bar, to make their execution more user friendly. Alternatively the prepro.apr project can be downloaded and opened in ArcView directly. In the prepro.apr project the scripts are already compiled and tied to buttons on the view button bar.

5.5.3. Starting the System.

The system is started by executing the HECPREPRO script either from the script editor or a button in the view button bar. When executing the script the input themes have to be active. This is done by clicking on the theme in the legend while holding down the 'Shift' key. The system will examine the input themes and identify which theme represents which feature. See Section 5.2 for a detailed description of the input data requirements. Table 5.25 lists the input themes and identification criteria.

Input Theme	**Required**	**Identification Criteria**
Stream	yes	line feature theme
Subbasin	yes	polygon feature theme
Stream Location	no	point feature theme
Elevation Grid	no	grid theme

Table 5.25. Input Themes.

The system then prompts the user to supply the run control parameters. Run control parameters are listed in Table 5.26.

Variable	Description	Default
attrib	Transfer attributes from the input data to the symbolic data layers and the HEC-HMS Basin file? (yes/no, see Section 5.5.4)	no
hmsmode	Where should the HEC-HMS Basin file be saved to? (default, path)	default
tol	Tolerance. (see Section 5.5.5)	10
oblevel	User Observation Level. (see Section 5.5.6)	2

Table 5.26. Run Control Parameters.

5.5.4. Working with Attributes.

Attributes, like SCS curve number for subbasins or slopes for reaches can be transferred from the input data to the HEC-HMS Basin file. For that, each hydrologic element type has an attribute transfer table associated with it.

Sample attribute transfer tables can be downloaded from the same location as the program files as described in Section 5.5.1 (see Table 5.28 for the names). A few examples are presented later in this section. The attribute transfer table specifies the name of the field in the input data, the name of the field in the output data and the transfer mode to be used. The attribute transfer modes are listed in Table 5.27

Value	Description
0	No transfer
1	Transfer, reach attribute total
2	Transfer, reach attribute simple average
3	Transfer, reach attribute length weighted average

Table 5.27. Attribute Transfer Modes.

The tables are identified based on their names as listed in Table 5.28.

Hydrologic Element Type	Attribute Transfer Table Name
Subbasin	hecsub.dbf
Reach	hecreach.dbf
Junction	hecjunct.dbf
Diversion	hecdiv.dbf
Reservoir	hecres.dbf
Source	hecsourc.dbf
Sink	hecsink.dbf

Table 5.28. Attribute Transfer Table Names.

The structure of the attribute transfer tables is identical. Each table has to be structured as shown in Table 5.29.

Field Name	Field Type	Field Length	Description
hmsfield	string	32	name of field in HEC-HMS Basin file.
gisfield	string	32	name of field in input layer attribute table
transfer	number	1	transfer mode (see Table 5.27)

Table 5.29. Attribute Transfer Table Structure.

5.5.4.1. Subbasin Elements.

Attributes of the subbasin polygon in the input data can be transferred to the subbasin point in the symbolic point shape file (see Section 5.4.5) and the subbasin element in the HEC-HMS Basin File. The field name from the input theme is used in the symbolic point shape file. The field name for the HEC-HMS Basin file is specified in the subbasin attribute transfer table. Table 5.30 lists an example subbasin attribute transfer table.

Hmsfield	**Gisfield**	**Transfer**
Label X	x_coord	1
Label Y	y_coord	1
Area	area	1
LossRate	lossrate	1
Percent Impervious Area	pia	1
Initial Loss	iniloss	1
Constant Loss Rate	conloss	1
Transform	trans	1
SnyderTp	snytp	1
SnyderCp	snycp	1
Baseflow	baseflow	1
Recession Factor	recfact	1
Flow / Area Ratio	fta	1
Flow to Peak Ratio	ftp	1

Table 5.30. Example Subbasin Attribute Transfer Table.

5.5.4.2. Reach Elements.

Attributes of the stream lines in the input data can be transferred to the reach line in the symbolic line shape file (see Section 5.4.4) and the reach element in the HEC-HMS Basin File. The field name from the input theme is used in the symbolic line theme. The field name for the HEC-HMS Basin file is specified in the reach attribute transfer table. Table 5.31 lists an example reach attribute transfer table and Figure 5.6 illustrates the methodology.

Hmsfield	**Gisfield**	**Transfer**
Description	desc	1
Label X		0
Label Y		0
Route	route	1
Muskingum K	muskk	3
Muskingum X	muskx	3
Muskingum Steps	muskst	1

Table 5.31. Example Reach Attribute Transfer Table.

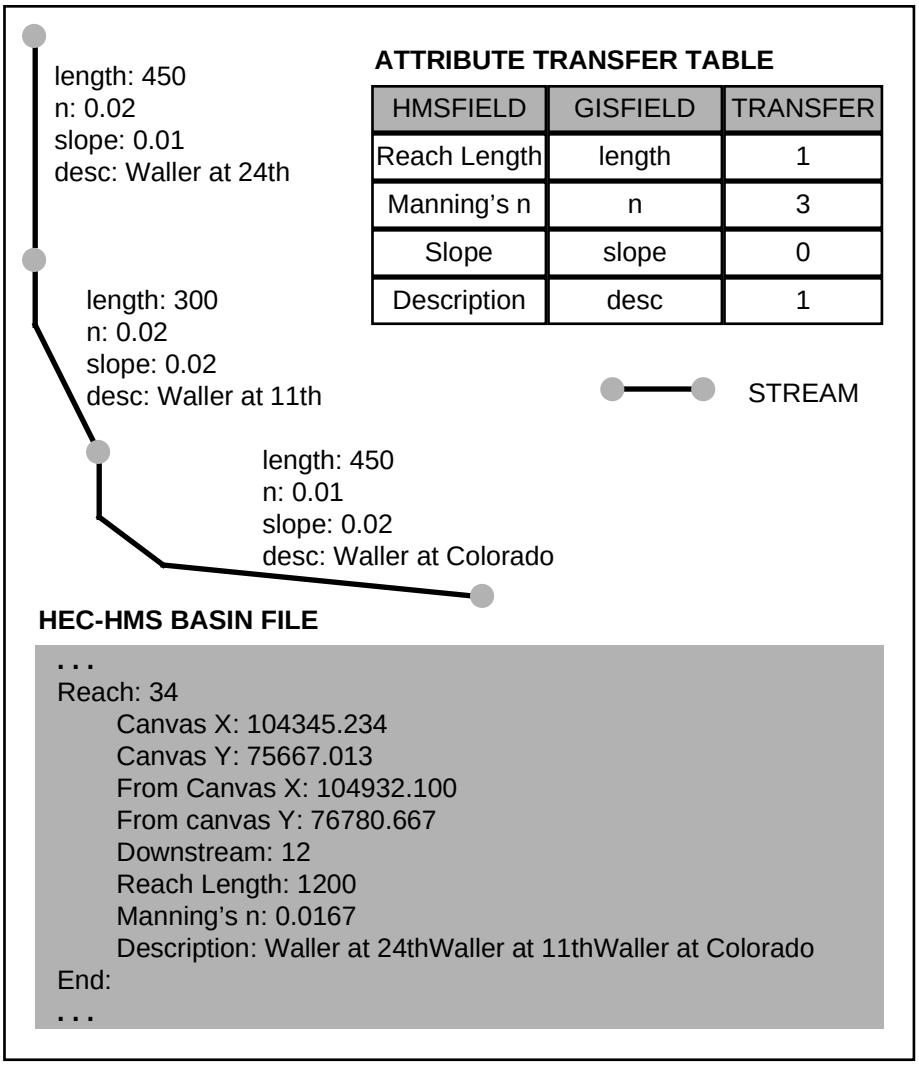

Figure 5.6. Reach Attribute Transfer Methodology.

For attributes that are transferred with the length weighted average option the length of the stream lines is determined from the coordinate list of the line.

5.5.4.3. Junction, Diversion, Reservoir, Source and Sink Elements.

If a stream location theme was specified, attributes of the point/node in the input data can be transferred to the element point in the symbolic point shape file (see Section 5.4.5) and the element in the HEC-HMS Basin File. The field name from the input theme is used in the sym point theme. The field name for the HEC-HMS Basin file is specified in the element attribute transfer table. Table 5.32 lists an example junction attribute transfer table.

Hmsfield	**Gisfield**	**Transfer**
Label X	x_coord	1
Label Y	y_coord	1
Observed Hydrograph Pathname	obhpath	1

Table 5.32. Example Junction Attribute Transfer Table.

5.5.5. Specifying Tolerance.

The tolerance specifies the spatial accuracy of the analysis. The tolerance is a distance in map units that should be set larger than the accuracy of the input data. Points that are distance less than the tolerance apart are considered to be connected. This is illustrated in Figure 5.7.

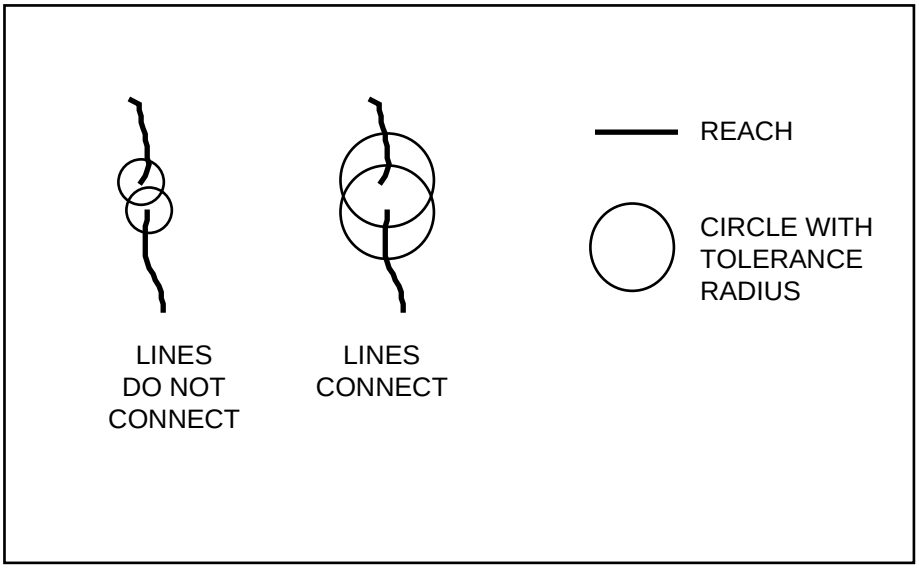

Figure 5.7. Tolerance Illustration.

Lines that are smaller than (half) the tolerance distance are also neglected from the analysis. This can serve as an alternative to the 'node snapping' option in the ARC/INFO implementation. Figure 5.8 illustrates this concept.

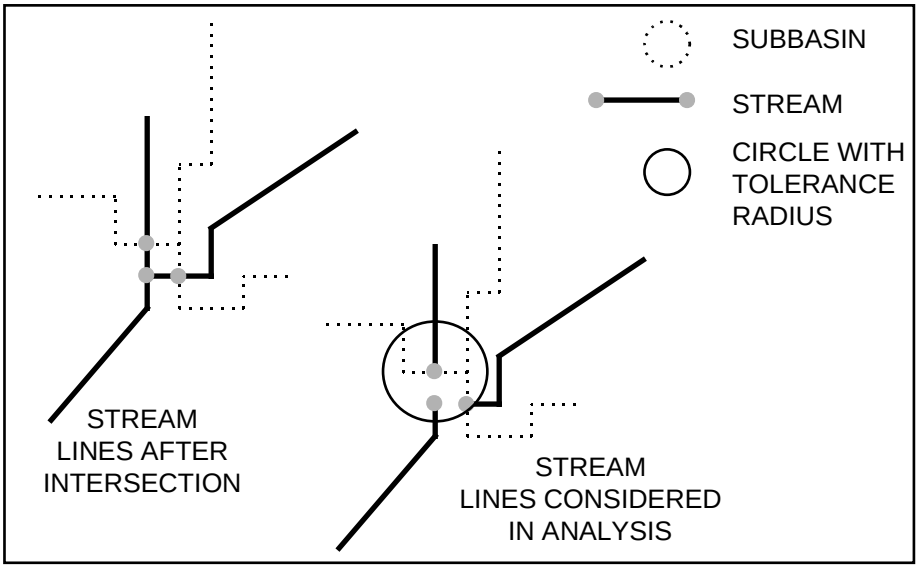

Figure 5.8. Using Tolerance as Node Snapping Alternative.

5.5.6. Specifying User Observation Level.

The user observation level controls the amount of information communicated to the user during program execution. The higher the user observation level the more information is displayed and the slower the program runs. Table 5.33 lists the user observation levels.

Value	Description
0	Not used.
1	Status bar is updated periodically.
2	Status bar is updated periodically and themes are added to the view and the display is redrawn at the end of each step.
3	Same as 2 except that the view is updated after each feature is added.
4	Debug mode.

Table 5.33. User Observation Level Description.

5.5.7. Viewing Output.

5.5.7.1. HEC-HMS Basin File.

The HEC-HMS Basin file is in ASCII format and can be viewed with any text editor or word processor.

5.5.7.2. Hydrologic and Symbolic Line and Point Shape File.

The hydrologic and symbolic line and point shape files (see Section 5.4) are automatically added to the view containing the input data. The HECLEGEND program can be used to set the legend of the themes. This program is also called from HECPREPRO at user observation levels of 2 or higher during program execution. Table 5.34 lists the colors assigned to each hydrologic element type.

Hydrologic Element Type	**Color**
None	black
Subbasin	gray
Subbasin Outlet	cyan
Reach	green
Junction	dark green
Diversion	magenta
Reservoir	blue
Source	light green
Sink	red

Table 5.34. Colors Assigned to Hydrologic Element Types by HECLEGEND.

6. APPLICATIONS OF THE HEC-PREPRO SYSTEM.

6.1. Introduction

At the time of this study there were several ongoing research projects at The University of Texas at Austin that used GIS as an analysis platform. One product of those studies is usually a digital description of a watershed suitable for input to HEC-PREPRO. Four of those studies were chosen as test cases for HEC-PREPRO. A brief description of the input data, HEC-PREPRO application and output data are given for each test case.

6.2. Niger River Watershed.

6.2.1. Input Data.

The Niger River watershed was studied using GIS by Olivera et al. (1995). They describe the region as follows:

> The Niger River basin is a 2.3 million square kilometer area found between 5° N and 23° N of latitude, and 12° W and 17° E of longitude. The river rises in Guinea and flows for about 4,200 Km through Mali, Niger and Nigeria before reaching the Atlantic Ocean. Its main tributary, the Benue River, flows west from Cameroon and joins the Niger at Lokoja, Nigeria. In general, the basin landscape shows a great variability with desertic areas in the northern part and tropical areas in the south. The Inner Delta, just upstream of the city of Tombouctou in

> Mali, is a 150,000 Km^2 almost flat area with extremely high water losses due to evaporation, which makes it particularly difficult to model. Precipitation in the basin is also very variable, and it ranges from 2,700 mm/yr close to the river mouth to almost none in the desertic parts.

Stream and subbasin lines were delineated from an elevation grid with cell size of 1,000 meters using ARC/INFO Grid routines. Stream junctions were chosen as the default location for subbasin outlets. There are 167 subbasins with an average area of 14,000 km^2. Figure 6.1. shows the delineated stream and subbasin lines of the watershed. The Atlantic ocean is shaded gray for orientation.

FIGURE 6.1. BLANK PAGE.

6.2.2. Application

This application led to the development of the node snapping option discussed in Section 4.5.4.3. As discussed in that section, vectorization of grid data can create small scale geographic errors. If the subbasin outlets are located at stream junctions the geographic error can introduce a hydrologic error. The node snapping option was used in this application with a snapping tolerance of 3,000 meters.

The watershed outlet node (sink) was located in a cell of the elevation grid which was not valued. This caused an error during HEC-PREPRO execution. To solve the problem the node was manually moved to an adjacent cell which was valued. Alternatively the cell could have been valued manually. Table 6.1 lists the run control parameters used for the application.

Run Control Parameter	Value
Stream Coverage	ngst
Subbasin Coverage	ngsub
Elevation Grid	ngelev
User Observation Level	0
Attribute Collection?	Y
Clipped?	N
Node Snapping?	Y
Snapping Tolerance	3000
Name of General Text File	gen.txt
DXF File?	N
Name of DXF File	N/A
HMS Basin File?	Y
Name of HMS Basin File	hmsbasin.txt

Table 6.1. Run Control Parameters for Niger River Watershed Application.

6.2.3. Output Data.

The resulting HEC-HMS Basin file has 374 hydrologic elements: 167 subbasins, 103 reaches, 103 junctions and 1 sink. Figure 6.2. shows the watershed displayed with a hydrologic data structure and Figures 6.3 and 6.4 show the watershed displayed in HEC-HMS.

FIGURE 6.2. BLANK PAGE.

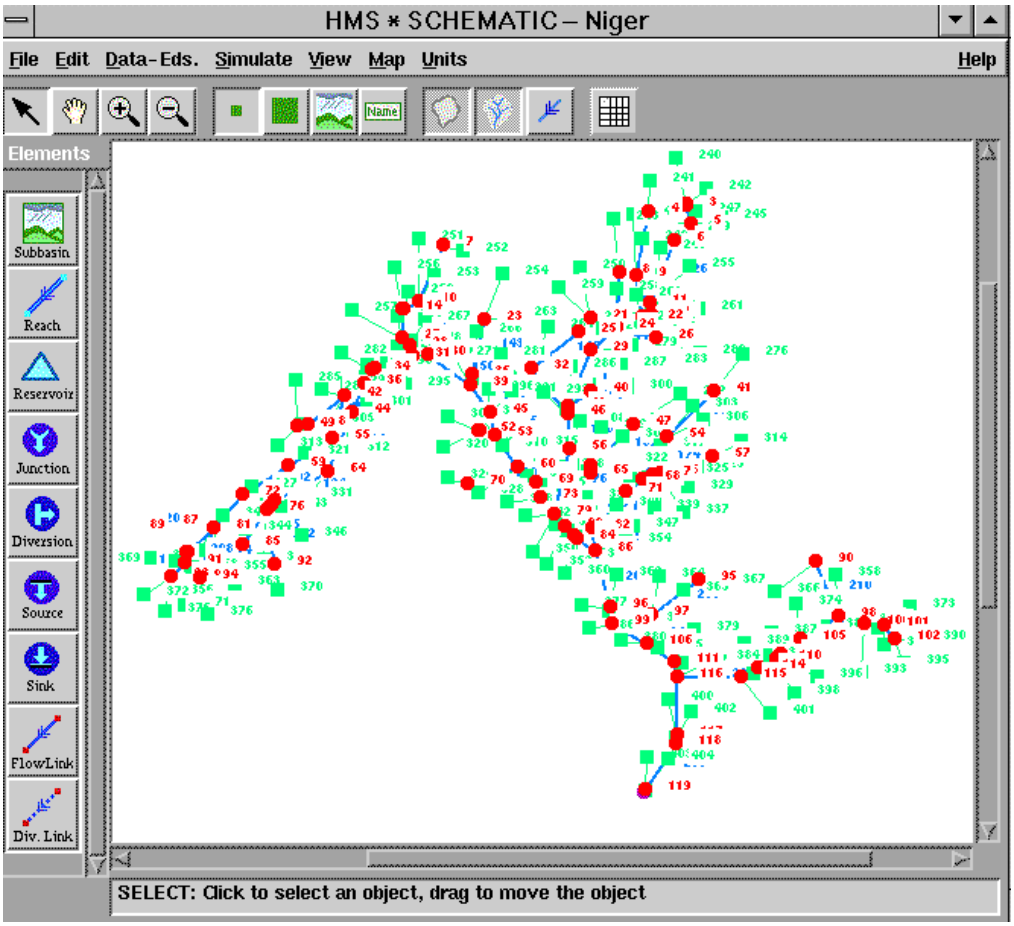

Figure 6.3. Screen Capture of HEC-HMS with the Entire Niger River Watershed Displayed.

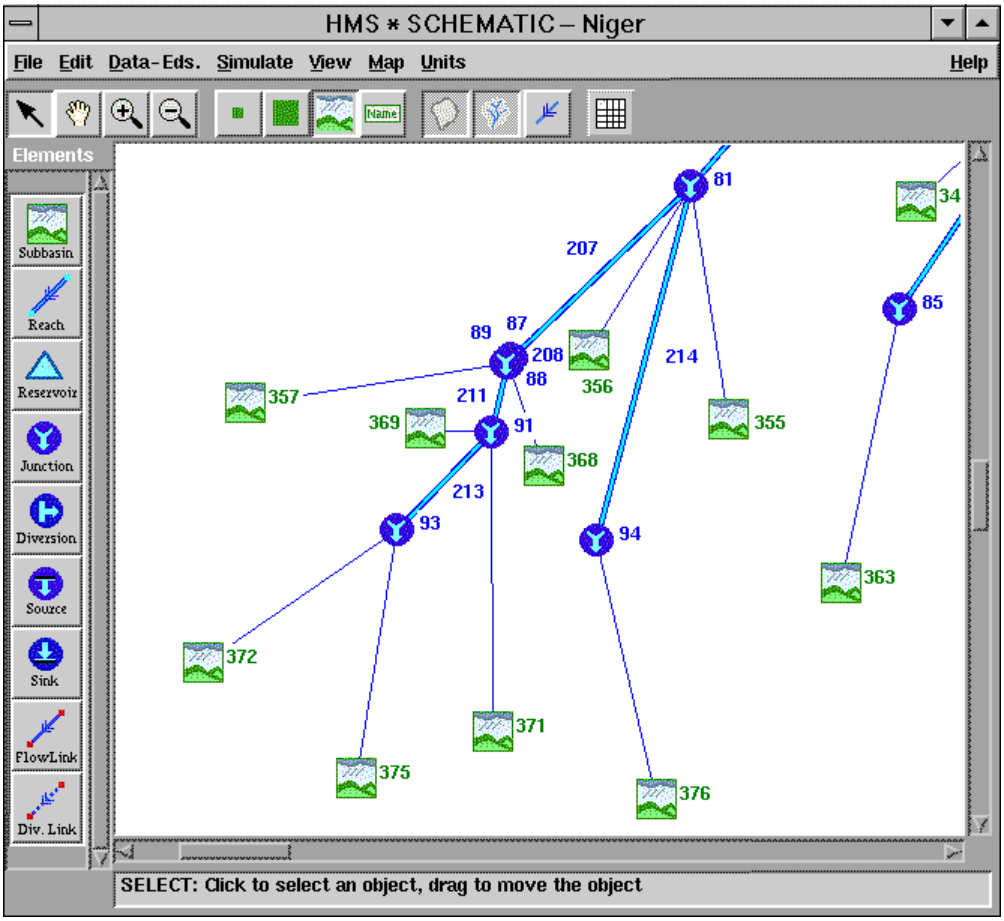

Figure 6.4. Screen Capture of HEC-HMS Zoomed into the Subbasin Draining to the Koulikoro Gaging Station.

6.3. State of Texas Watersheds.

6.3.1. Input Data.

The watersheds of the State of Texas were studied by Reed et al. (1997). A brief introduction to the study is given by them as follows:

> Water availability is critical to the economy in the state of Texas. Numerous reservoirs and conveyance structures have been constructed across the State to meet the water supply needs of farmers, municipalities, industries, and power generating facilities. Despite this extensive water management system, water supply remains a concern because of increasing populations and uncertainties about climate stability.

Stream and subbasin lines were delineated from an elevation grid with approximate cell size of 500 meters using ARC/INFO Grid routines. Subbasins were delineated from stream gaging stations. There are 30 watersheds subdivided into 166 subbasins with an average area of 6,600 km^2. Figure 6.5 shows the delineated stream and subbasin lines of the watersheds. The State of Texas area is shaded gray for orientation.

FIGURE 6.5. BLANK PAGE.

6.3.2. Application.

The delineation of stream and subbasin lines from an elevation grid with ARC/INFO Grid routines causes small scale geographic errors that caused problems for the Niger River application, as described in the previous section. The difference between this input data and that for the Niger River watershed is that here the subbasins are delineated from stream gaging stations. Subbasin outlets were therefore not located at stream junctions and geographic errors introduced by the vectorization of the grid data did not introduce any hydrologic errors. Table 6.2 lists the HEC-PREPRO run control parameters used for the application.

Run Control Parameter	Value
Stream Coverage	txst
Subbasin Coverage	txsub
Elevation Grid	txelev
User Observation Level	0
Attribute Collection?	N
Clipped?	N
Node Snapping?	N
Snapping Tolerance	N/A
Name of General Text File	gen.txt
DXF File?	N
Name of DXF File	N/A
HMS Basin File?	Y
Name of HMS Basin File	hmsbasin.txt

Table 6.2. Run Control Parameters for State of Texas Watersheds Application.

6.3.3. Output Data.

The resulting HEC-HMS Basin file has 572 hydrologic elements: 166 subbasins, 188 reaches, 188 junctions and 30 sinks. Figure 6.6. shows the watersheds displayed with a hydrologic data structure.

FIGURE 6.6. BLANK PAGE.

6.4. 1993 Midwest Flood Watershed.

6.4.1. Input Data.

The 1993 Midwest Flood watershed was studied by Mizgalewicz and Maidment (1996). A brief description of the project is given by them as follows:

> In 1993 a large flood occurred in the Midwest of the United States covering all of Iowa and portions of surrounding states. Major damage was caused by overland flow throughout the region, and serious levee breaches along the Mississippi and Missouri Rivers and their tributaries let to inundation of surrounding farmlands and urban areas. Shortly after the event, President Clinton appointed the Scientific Assessment and Strategy Team (SAST) to study the flood and make policy recommondations as to how the damage from future floods in this region could be mitigated. There were two general policy questions: what was the effect of levee failures on flood inundation, and what was the effect of regional drainage of wetlands on the volume of flood waters discharge onto downstream communities. The first question, hydraulic in nature, was analyzed using water surface profile models along the main rivers. The second question, hydrologic in nature, was difficult to answer because of the huge region affected by the flood, some 700,000 km^2. Most of the hydrologic studies initiated by the SAST project dealt with much smaller watersheds within the region but it was difficult to generalize their conclusions because there was no convenient hydrologic model of the whole flooded region available at that time.
>
> As a later part of the SAST study a daily water balance of the flood was constructed in a GIS environment at the Universities of Texas and Utah, in which the flooded region was divided into 261 subbasins using USGS stream gages as the outlet point of each subbasin (Mizgalewicz et al., 1997). The EPA River Reach File 1 (RF1) stream

network was embedded into a 500 meter digital elevation model of the basin, terrain analysis performed using ARC/INFO to recover the gridded stream network, and subbasins delineated for each of the 261 gage locations.

The average subbasin area was 2,700 km^2. Figure 6.7 shows the watershed described with a geographic data structure. The great lakes are shaded gray for orientation.

FIGURE 6.7. BLANK PAGE.

6.4.2. Application.

In the input data, the upstream end of several streams was located close to the watershed boundary. The result was that HEC-PREPRO interpreted those points as sources into the watershed. The stream lines at those locations were edited manually to correct the problem. The input data also had several mini-polygons at subbasin lines which were removed manually. Several stream lines were not pointing in the downstream direction. They were flipped manually. The run control parameters used for the application are listed in Table 6.3.

Run Control Parameter	Value
Stream Coverage	sastst
Subbasin Coverage	sastsub
Elevation Grid	sastelev
User Observation Level	0
Attribute Collection?	N
Clipped?	N
Node Snapping?	N
Snapping Tolerance	N/A
Name of General Text File	gen.txt
DXF File?	N
Name of DXF File	N/A
HMS Basin File?	Y
Name of HMS Basin File	hmsbasin.txt

Table 6.3. Run Control Parameters for 1993 Midwest Flood Watershed Application.

6.4.3. Output Data.

The resulting HEC-HMS basin file has 1,076 hydrologic elements: 5 sources, 261 subbasins, 402 junctions, 407 reaches and 1 sink. Figure 6.8 shows the watershed described with a hydrologic data structure and Figures 6.9 and 6.10 show the watershed displayed in HEC-HMS.

FIGURE 6.8. BLANK PAGE.

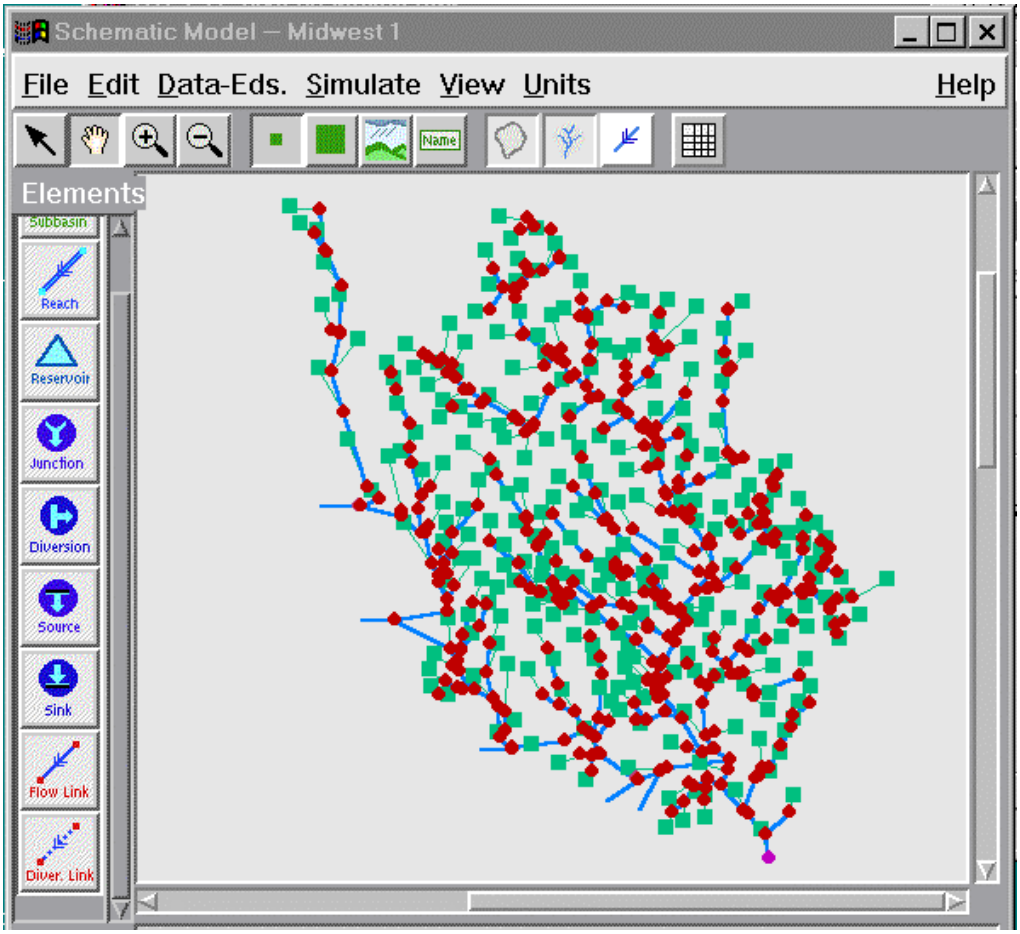

Figure 6.9. Screen Capture of HEC-HMS with the Entire 1993 Midwest Flood Watershed Displayed.

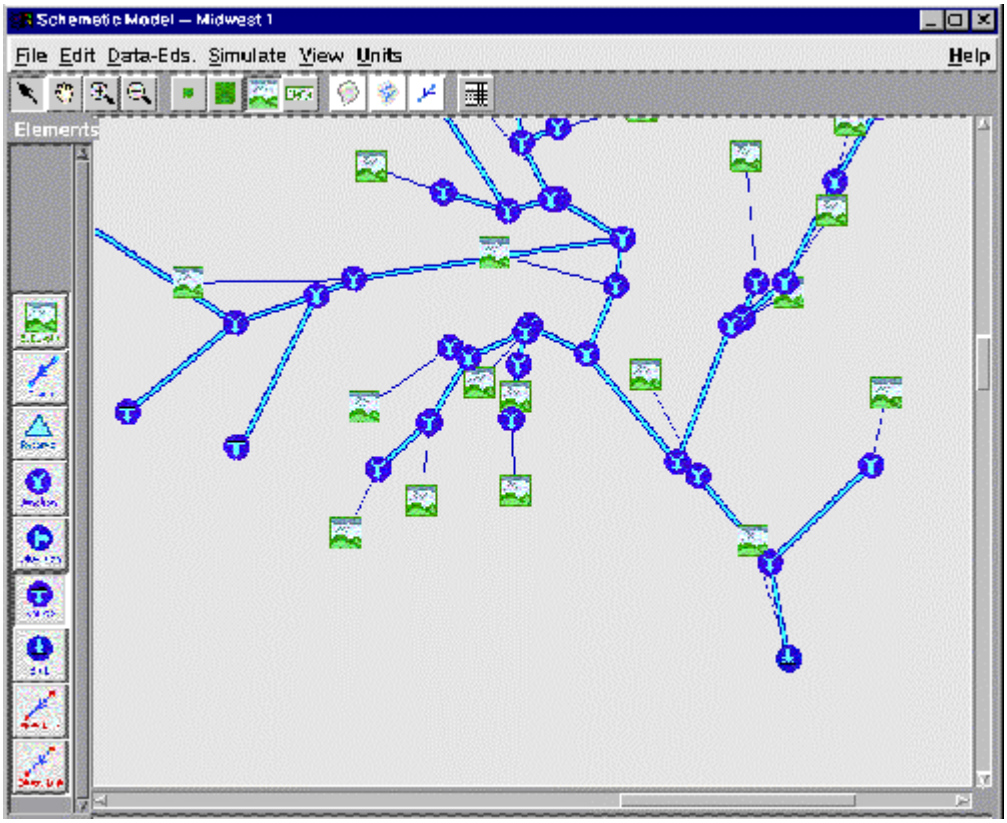

Figure 6.10. Screen Capture of HEC-HMS Zoomed in to the Junction of the Missouri and Mississippi Rivers.

6.5. City of Austin, Texas Watersheds.

6.5.1. Input Data.

The surface water hydrology of the City of Austin, Texas was studied by Olivera, et al. (1996). The stream and subbasin lines were delineated with ARC/INFO grid routines using a grid size of approximately 100 meters. Subbasins were delineated from stream junctions. There are 133 subbasins with an average area of 17 km^2. Figure 6.11 shows the delineated stream and subbasin lines of the watersheds. Lakes of the area are shaded gray for orientation.

FIGURE 6.11. BLANK PAGE.

6.5.2. Application.

The input data contained small scale errors at the subbasin outlets similar to those in the Niger River watershed input data. HEC-PREPRO was therefore run with the node snapping option with a snapping tolerance of 1000 ft (~300m). Note that the snapping tolerance is smaller than the one used for the Niger River watershed (3000 m). The spatial scale of error is controlled by the grid cell size used for the stream and subbasin delineation. Since the grid cell size for this application is ten times smaller than the cell size for the Niger River watershed, the spatial scale of error is ten times smaller. Note that for both cases a snapping tolerance of three times the cell size worked well for both applications. The optimal node snapping tolerance is determined by trial and error. Several stream lines had to be edited manually. The stream line representing the Colorado River at Lake Travis, for example, was extended across the watershed boundary so that HEC-PREPRO would recognize it as a source. Table 6.4 lists the run control parameters used for the application.

Run Control Parameter	Value
Stream Coverage	ausst
Subbasin Coverage	aussub
Elevation Grid	auselev
User Observation Level	0
Attribute Collection?	N
Clipped?	N
Node Snapping?	Y
Snapping Tolerance	1000
Name of General Text File	gen.txt
DXF File?	N
Name of DXF File	N/A
HMS Basin File?	Y
Name of HMS Basin File	hmsbasin.txt

Table 6.4. Run Control Parameters for City of Austin, Texas Watersheds Application.

6.5.3. Output Data.

The resulting HEC-HMS basin file has 270 hydrologic elements: 1 source, 133 subbasins, 68 junctions, 69 reaches and 1 sink. Figure 6.12 shows the watersheds described with a hydrologic data structure and Figures 6.13 and 6.14 show the watersheds displayed in HEC-HMS.

FIGURE 6.12. BLANK PAGE.

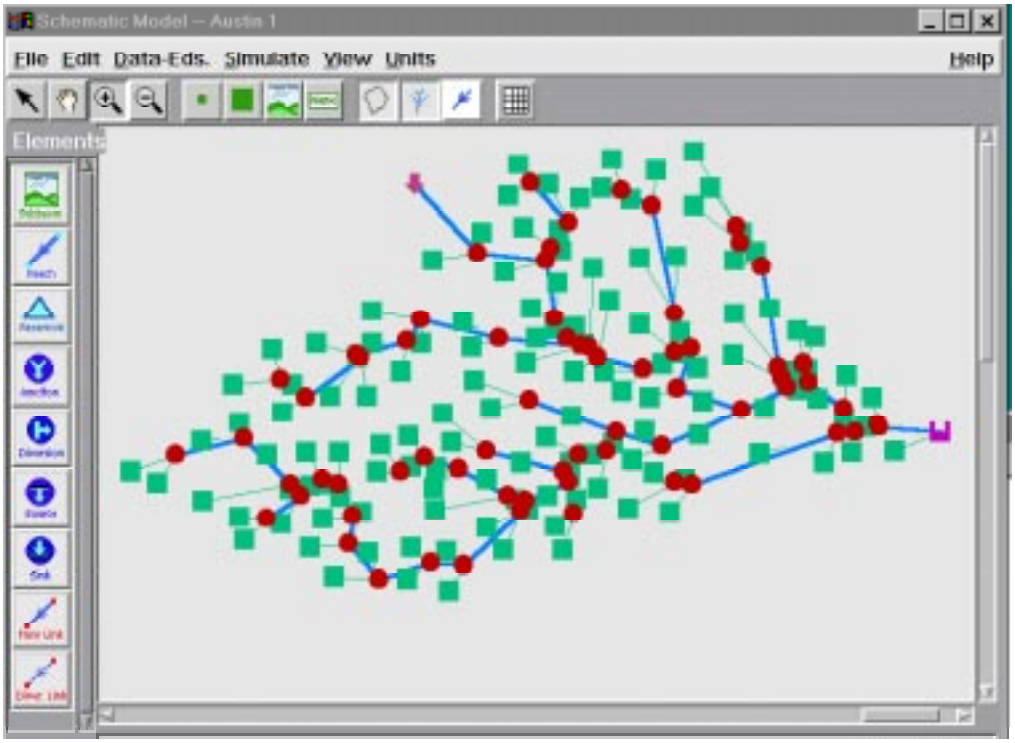

Figure 6.13. Screen Capture of HEC-HMS with all City of Austin, Texas Watersheds Displayed.

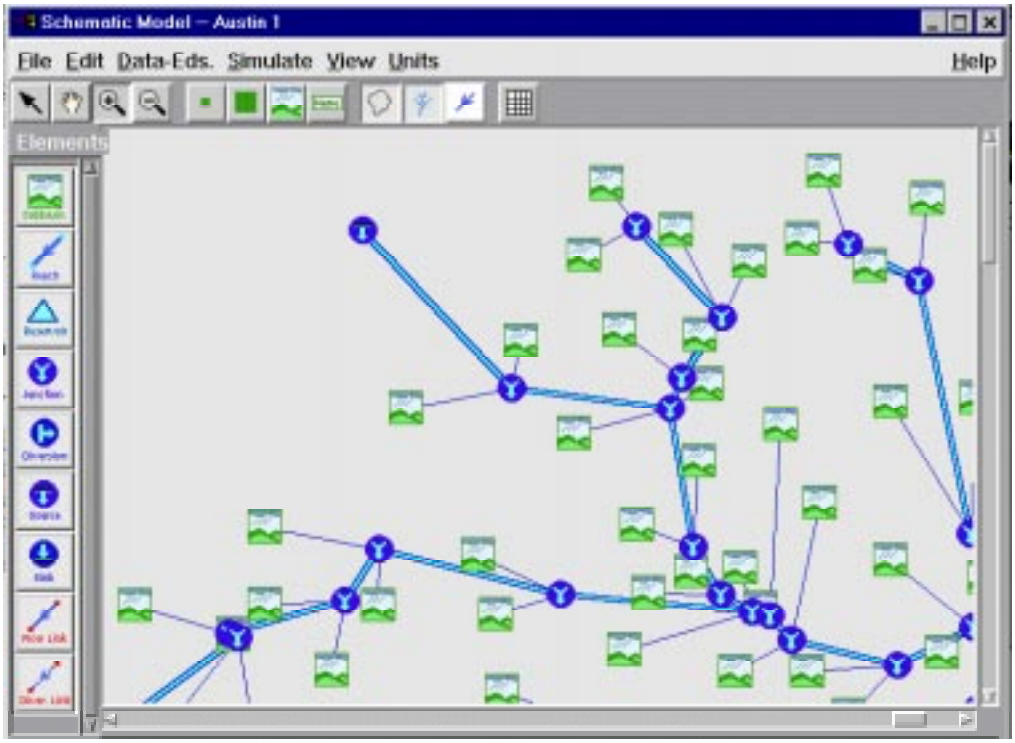

Figure 6.14. Screen Capture of HEC-HMS Zoomed in to the Outlet of Lake Travis.

6.6. Summary of HEC-PREPRO Applications.

Table 6.5 summarizes the range of hydrologic elements and Table 6.6 summarizes spatial statistics or the applications.

	Number of Elements					
Region	**Total**	**Subbasin**	**Reach**	**Junction**	**Source**	**Sink**
Niger	374	167	103	103	0	1
Texas	572	166	188	188	0	30
Midwest	1,076	261	402	407	5	1
Austin	270	133	68	69	1	1

Table 6.5. Summary of Hydrologic Element Statistics for Applications.

Region	**Region Area [km^2]**	**Subbasin Area [km^2]**	**Grid Cell Size [m]**
Niger	2,300,000	14,000	1,000
Texas	1,100,000	6,600	500
Midwest	700,000	2,700	500
Austin	2,300	17	100

Table 6.6. Summary of Spatial Statistics for Applications.

7. CONCLUSIONS.

This chapter summarizes the conclusions drawn from the study. The conclusions are categorized into general conclusions, followed by conclusions related to the methodology, implementations and applications of the system. Then some thoughts on future work are presented.

7.1. General.

A methodology, called HEC-PREPRO, for automating a part of the lumped parameter hydrologic modeling process has been developed, implemented and tested. This methodology presents the potential for significant time savings in the hydrologic modeling process. It makes hydrologic modeling more feasible. The hydrology of larger regions can be modeled with greater detail.

7.2. Methodology.

A method, HEC-PREPRO, for converting watershed data from a geographic to a hydrologic data structure has been developed. The geographic data structure describes the watershed by locating its features in space. For HEC-PREPRO this consists of the location of streams, subbasins, as well as elevations. The hydrologic data structure describes the watershed by defining the hydrologic function of components of the watershed and how those components connect. For HEC-PREPRO this consists of seven classes of hydrologic elements: subbasins, sources, reaches, junctions, reservoirs, diversions and sinks.

The procedure, HEC-PREPRO, is designed for implementation in a geographic information system (GIS) environment. The input data consists of digital vector data layers of streams and subbasins and a digital raster data layer of elevations. The procedure is presented in general, yet distinct steps so that it may be implemented in any GIS environment. The GIS environment needs to be capable of managing vector data and to provide for a programming interface if the procedure is to be automated. The output data consists of the definition and connection of hydrologic elements presented in

the form of a stick diagram type digital vector data layer laid over the input data, or in the form of an HEC-HMS Basin file, an input file to a lumped parameter hydrologic modeling software program.

The HEC-PREPRO methodology heavily relies on a node-line topology. This topology classifies each node on the stream network into one of ten types based on the number and orientation of adjacent stream lines. For example: A node that has multiple lines upstream and one line downstream is classified as a junction. This type of topology is termed hydro-topology, which describes the hydrologic function of each node on a micro scale. Functionality for developing or building hydro-topology is usually not provided for directly in the GIS environment which means it has to be programmed manually.

7.3. Implementations.

The procedure, HEC-PREPRO, has been implemented in two different geographic information system (GIS) environments: ARC/INFO and ArcView. Both systems are developed by the Environmental Systems Research Institute, Inc. (ESRI), Redlands, California. In ARC/INFO, the implementation was done with the Arc Macro Language (AML), a simple macro language that allows for the definition of simple variables and loops. In ArcView the implementation was done with Avenue, an ArcView interpreted, object oriented programming language. A library of object classes is available in Avenue, but new classes cannot be defined and existing class definitions cannot be modified.

The ArcView Avenue programming language is more efficient than the ARC/INFO AML programming language. The HEC-PREPRO implementation in Avenue requires approximately three thousand lines of code whereas the implementation in AML requires approximately six thousand lines of code. It took also considerably less time to program the Avenue implementation, but some of that difference is due to the fact that the Avenue version was programmed after the AML version. The programming

of the Avenue version benefited from the experience gained during the programming of the AML version.

The ArcView Avenue programming language is more flexible and contains more functionality than ARC/INFO AML. For example: Avenue allows for random access to variable arrays and matrices, whereas in AML those data sets have to be processed sequentially. As a result the ArcView implementation of HEC-PREPRO runs considerably faster.

The ARC/INFO GIS system has more functionality built in than does ArcView. Several topology, network and data manipulation functions are needed by HEC-PREPRO, including the building of line-node topology, tracing along lines and intersecting of two data layers. Those functions are available in ARC/INFO and can be called directly from AML. ArcView does not support many of those functions which means they had to be programmed into the Avenue code.

The ArcView GIS system provides more flexibility than the ARC/INFO GIS system. For example: In ArcView access to the geographic definition of points, lines and polygons is provided directly through shapes

that can be accessed by the Avenue programming language. Properties of shapes can be examined and modified easily. In ARC/INFO the geographic definition of features are modified mostly through functions. Direct modification of the geographic definition of features is possible through the ArcEdit module, but the functions available there are limited.

The HEC-PREPRO procedure is designed to be run without user interaction. For troubleshooting and demonstration purposes it was found useful to communicate information to the user at various points in the program execution. This led to the development of a user observation level concept. The user observation level defines the amount of information conveyed to the user during program execution. A high user observation level makes the program stop many times and prompt the user to 'hit any key to continue'. A low user observation level will cause the program to run with communicating as little information as possible to the user. In the ARC/INFO implementation, for example, a user observation level of zero is designed for background running.

7.4. Applications.

The HEC-PREPRO implementations have been applied to several watersheds of different sizes. Four of those applications are presented. They are the Niger River watershed, the State of Texas Watersheds, the 1993 Midwest Flood watershed and the City of Austin Watersheds. The regions range in size from 2,300 to 2,300,000 km^2. The input data consisted of streams and subbasins delineated with grid routines as well as standard data sets like the EPA's River Reach File 1 (RF1). Once processed the HEC-HMS Basin files of the regions ranged from 270 to 1,076 hydrologic elements.

These applications illustrated that the input data needs to be free of hydrologic errors. Hydrologic errors are different from geographic or topologic errors. Examples of geographic errors are streams shifted or stream gaging stations located off streams. Examples of topologic errors are watershed boundaries that do not close or two streams located on top of each other. Examples of hydrologic errors are streams zigzagging across subbasin boundaries or stream lines pointing upstream. A geographic or topologic error can constitute a hydrologic error as well, but does not have to do so.

Small geographic errors almost always exist. To prevent those errors constituting topologic or hydrologic errors, a tolerance distance can be defined to overlook those errors. In ARC/INFO this is termed the fuzzy tolerance. Consider for example the line-node topology building function. When the distance between the end points of two lines is within the fuzzy tolerance the lines are assumed to connect and are assigned the same node number. A geographic error does not constitute a topologic error in this case. Shape files, used by ArcView, do not possess such an attribute. However, since the topologic functions are programmed manually in HEC-PREPRO, the tolerance distance has been programmed into the implementation in a similar way to the fuzzy tolerance.

The ARC/INFO GIS environment is capable of handling larger files than the ArcView environment. Both the ARC/INFO and ArcView implementations of HEC-PREPRO were able to process the Niger River watershed, consisting of 270 hydrologic elements. Only the ARC/INFO implementation was able to handle the 1993 Midwest Flood watershed, consisting of 1,076 hydrologic elements. The ArcView implementation was not able to process this watershed running on a PC or UNIX machine.

7.5. Future Work

In the future the HEC-PREPRO implementations should be tested on more complex larger regions containing reservoirs and diversions. The system does support those features, but it was never used on regions containing those features, except for the development data.

The existing implementations of the procedure should be used in the actual hydrologic modeling process. This usage will define future needs and direction for HEC-PREPRO.

APPENDIX A. ARC/INFO AML PROGRAMS.

A.1. Main Program (hecprepro.aml).

```
/*
/*-----------------
/*-----------------
/*--- HECPREPRO ---
/*-----------------
/*-----------------
/*
/*----------------------------
/*--- Creation Information ---
/*----------------------------
/*
/*Name: hecprepro.aml
/*Version: 3.1
/*Date: 08/19/96
/*Author: Ferdi Hellweger
/*        Center for Research in Water Resources
/*        The University of Texas at Austin
/*        ferdi@crwr.utexas.edu
/*
/*----------------------------
/*--- Purpose/Description ---
/*----------------------------
/*
/*HECPREPRO is a GIS preprocessor for the Hydraulic Engineering Center's (HEC)
/*Hydrologic Modeling System (HMS).  HMS is currently being developed HEC as
part
/*of the NexGen program of research.  The purpose of HECPREPRO is to summarize
data
/*from a GIS system so that they can be used as input for HMS.  HECPREPRO
takes a
/*stream coverage, a subbasin coverage and an elevation grid as input data.
The output
/*data consists of an HMS basin file, a general text file, a DXF file and a
'hydrologic'
/*and a symbolic coverage. The system is written in ARC/INFO's Arc Macro
Language (AML)
/*and is designed to run on a UNIX platform.
/*
/*---------------------
/*--- General Notes ---
/*---------------------
/*
/*1. Sponsor
/*This work was sponsored by the Hydrologic Engineering Center,
/*US Army Corps of Engineers, Davis, Calif. and supervised by
/*David R. Maidment.
/*
/*2. Previous Work
/*The methodolgy used here was adopted with modifications from the
/*report by David R Maidment entitled "Developing a Watershed Data
/*Structure" prepared for the Hydrologic Engineering Center, US Army
/*Corps of Engineers, Davis, Calif., under Contract DACW05-92-P-1864,
/*dated April 9, 1993.
/*
/*3. Feature Summary
/*The geographic features are summarized as follows:
/*
/*Subbasin - hecid, x-coord, y-coord, cenz, area, means,
/*           medians, b-coord, f-coord, fupz, flength, flength2
/*           hecdnid
/*Source - hecid, x-coord, y-coord, hecdnid
/*Reach - hecid, length, slope, hecupid, hecdnid
/*Junction - hecid, x-coord, y-coord, hecupid(s), hecdnid
/*Reservoir - hecid, x-coord, y-coord, hecupid(s), hecdnid(s)
/*Diversion  - hecid, x-coord, y-coord, hecupid, hecdnids
/*Sink - hecid, x-coord, y-coord, hecupid(s)
/*
/*If the %attrib% variable is set to 'no' attributes are not
```

```
/*summarized and the output file will only contain the connectivity
/*attributes hecid, hecupid and hecdnid.
/*
/*4. Prior Clipping or Intersecting
/*If the stream coverage was clipped or intersected with polygons
/*from the subbasin coverage the program has to use a slightly
/*different method to identify the channel system.  The %clipped%
/*variable controls this.
/*
/*5. Node Snapping
/*Nodes from the stream coverage can be snaped to nodes from the
/*subbasin coverage.  This is useful for data created with GRID
/*routines.  The %snapit% variable controls whether snapping will
/*be done and the %snaptol% variable controls the tolerance used
/*which should be set with grid size in mind.
/*
/*------------------
/*--- Input Data ---
/*------------------
/*
/*1. Stream Coverage
/*-all arcs have to point downstream
/*-reservoirs are closed polys
/*
/*2. Subbasin Coverage
/*-all subbasins have to contain streams
/*
/*3. Elevation Grid
/*
/*-------------------
/*--- Output Data ---
/*-------------------
/*
/*hydrocov - working arc coverage
/*symcov - symbolic coverage of watershed
/*%outfile% - general text file
/*%hmsname% - hms basin file
/*%dxfname% - dxf background map
/*
/*------------------
/*--- Proceedure ---
/*------------------
/*
/*The proceedure of the program is based on establishing the
/*connectivity of the elements.  Attributes are collected and
/*summarized where applicable, usually where the information
/*is at hand.  Connectivity is based on the next downstream
/*element(s).
/*
/*Step 1. Data Set Up
/*Step 2. Establish Source, Sink and Subbasin Outlet Elements
/*Step 3. Establish Channel System
/*Step 4. Establish Channel Elements
/*Step 5. Calculate Connectivity
/*Step 6. Create Symbolic Coverage
/*Step 7. Relate Attributes to Symbolic Coverage
/*Step 8. Create Output
/*
/*------------------------
/*--- Related Programs ---
/*------------------------
/*
/*hecshell.aml - shell that runs hecprepro (arc)
/*hecpause.aml - user observation level pause (called)
/*hechydro.aml - re-displays hydrocov (arcplot)
/*hecsym.aml - re-displays symcov (arcplot)
/*
/*-------------------------
/*--- Programming Notes ---
/*-------------------------
/*
```

```
/*1. Hydrocov Working Attributes
/*The following attributes are assigned to features:
/*
/*HECID - Unique number identifying each hydrologic element
/*HECTYPE - Number identifying the hydrologic type as follows
/* -1 - unknown
/*  1 - subbasin outlet
/*  2 - dangling stream
/*  3 - junction
/*  4 - channel
/*  5 - subbasin
/*  6 - sink
/*  7 - source
/*  8 - diversion
/*  9 - not used
/* 10 - reservoir
/*HECUPID - ID of upstream element
/*HECDNID - ID of downstream element
/*
/*2. Major Variables
/*There are number of variable arrays and matrices that get added on
/*and used throughout the program.  They are:
/*
/*rnodetype%node% - real node type indexed on node (hydrocov)
/*hnodetype%node% - hecnode type indexed on node (hydrocov)
/*h#tnodes%node% - number of downstream arcs indexed on node (hydrocov)
/*h#fnodes%node% - number of upstream arcs indexed on node (hydrocov)
/*nodez%node% - node elevation indexed on node (hydrocov)
/*nhectype%node% - hectype indexed on node (hydrocov)
/*nhecid%node% - hecid indexed on node (hydrocov)
/*hectype%hecid% - hectype indexed on hecid
/*hecx%hecid% - x coordinate indexed on hecid
/*hecy%hecid% - y coordinate indexed on hecid
/*%n%hecupid%hecid% - hecid of nth upstream element
/*%n%hecdnid%hecid% - hecid of nth downstream element
/*hecid2%node% - hecid indexed on node (symcov)
/*
/*3. Cursors and Loops
/*Cursors and loops are numbered by step number followed by their
/*number. Loop starts and ends are indicated by ls and le respectively.
/*
/*4. User Observation Level
/*0 - no pause and no graphic display (for background running)
/*1 - pause at error
/*2 - pause at error and legends (default)
/*3 - first level debug
/*4 - second level debug
/*0.20, 0.30, etc. - makes first pause at Step 2, Step 3, etc.
/*9 - at the pause prompt quits instantly
/*
/*
/*---------------------------
/*---------------------------
/*--- Step 1. Data Set Up ---
/*---------------------------
/*---------------------------
/*
/*----------------------------------------
/*--- Manage Command Line Input (1.1.) ---
/*----------------------------------------
/*
/*command line input:
/*
/*streamcov - Stream Coverage (required)
/*subcov - Subbasin Coverage (required)
/*elevgrid - Elevation Grid (required)
/*.oblevel - User Observation Level
/*attrib - Attribute Collection?
/*clipped - Prior Clipping?
/*snapit - Node Snapping?
/*snaptol - Snapping Tolerance Distance
```

```
/*outfile - Name of Output File
/*dxffile - Create DXF File?
/*dxfname - Name of DXF File
/*hmsfile - Create HMS Basin File?
/*hmsname - Name of HMS Basin File
/*
/*--- get command line input ---
/*
&args streamcovin subcovin elevgrid .oblevel attrib clipped snapit ~
snaptol outfile dxffile dxfname hmsfile hmsname
/*
/*--- check for input error ---
/*
&sv error = no
&if ( [ length %streamcovin% ] = 0 ) &then
   &sv error = yes
&if ( [ length %subcovin% ] = 0 ) &then
   &sv error = yes
&if ( [ length %elevgrid% ] = 0 ) &then
   &sv error = yes
&if ( %error% = yes ) &then
   &do
   &type HECPREPRO:
   &type HECPREPRO: INPUT ERROR - USAGE:
   &type HECPREPRO:
   &type
   &type &r hecprepro <stream arc coverage> <subwatershed poly coverage> ~
<elevation grid> {user observation level} {attribute summary} {prior clipping}
~
{node snapping} {node snapping tolerance} {name of output file}
   &type
   &type HECPREPRO:
   &type HECPREPRO: (HECSHELL.AML is interactive)
   &type HECPREPRO:
   &return
   &end
/*
/*--- assign default to .oblevel variable if not specified ---
/*
&if ( [ length %.oblevel% ] = 0 ) &then
&sv .oblevel = 2
/*
/*--- assign default to attrib variable if not specified ---
/*
&if ( [ length %attrib% ] = 0 ) &then
&do
&sv attrib = yes
&end
/*
/*--- assign default to clipped variable if not specified ---
/*
&if ( [ length %clipped% ] = 0 ) &then
&do
&sv clipped = yes
&end
/*
/*--- assign default to snapit and snaptol variables if not specified ---
/*
&if ( [ length %snapit% ] = 0 ) &then
&do
&sv snapit = no
&sv snaptol = 0
&end
/*
/*--- assign default to outfile variable if not specified ---
/*
&if ( [ length %outfile% ] = 0 ) &then
&sv outfile = hecprepro.out
/*
/*--- assign default to dxffile variable if not specified ---
/*
```

```
&if ( [ length %dxffile% ] = 0 ) &then
&sv dxffile = no
/*
/*--- assign default to dxfname variable if not specified ---
/*
&if ( [ length %dxfname% ] = 0 ) &then
&sv dxfname = dxfout.dxf
/*
/*--- assign default to hmsfile variable if not specified ---
/*
&if ( [ length %hmsfile% ] = 0 ) &then
&sv hmsfile = no
/*
/*--- assign default to hmsname variable if not specified ---
/*
&if ( [ length %hmsname% ] = 0 ) &then
&sv hmsname = hmsbasin.txt
/*
/*-------------------------------------
/*--- Display Program Start Message ---
/*-------------------------------------
/*
&type HECPREPRO:
&type HECPREPRO: ------------------------
&type HECPREPRO: --- HECPREPRO START ---
&type HECPREPRO: ------------------------
&type HECPREPRO:
/*
/*-----------------------------
/*--- Set Up Error Handling ---
/*-----------------------------
/*
/*--- initiate error count variable ---
/*
&sv errors = 0
/*
/*-----------------------------------------------------
/*--- Make Working Copies of Input Data Sets (1.2.) ---
/*-----------------------------------------------------
/*
/*--- stream coverage ---
/*
&if [exists streamcov -cover] &then
kill streamcov all
copy %streamcovin% streamcov
/*
/*--- subbasin coverage ---
/*
&if [exists subcov -cover] &then
kill subcov all
copy %subcovin% subcov
/*
/*--- elevation grid ---
/*
/*data set is not modified
/*
/*---------------------------------------
/*--- Create Grid of Subbasins (1.3.) ---
/*---------------------------------------
/*
&if ( %attrib% = yes ) &then
&do
/*-ls0005
/*
/*--- kill previously existing subgrid ---
/*
&if [exists subgrid -grid] &then
kill subgrid all
/*
/*--- get cell size from elevation grid ---
/*
```

```
&describe %elevgrid%
&sv cellsize = %GRD$DX%
/*
/*--- create grid ---
/*
&type HECPREPRO:
&type HECPREPRO: Creating grid of subbasins
&type HECPREPRO:
/*
polygrid subcov subgrid
%cellsize%
y
/*-le0005
&end
/*
/*-----------------------------
/*--- Build Topology (1.4.) ---
/*-----------------------------
/*
build streamcov node
build streamcov line
/*
build subcov node
build subcov line
build subcov poly
/*
centroidlabels subcov
addxy subcov
/*
/*-----------------------------------------------
/*--- Add Working Items to INFO Files (1.5.) ---
/*-----------------------------------------------
/*
/*--- streamcov.nat ---
/*
additem streamcov.nat streamcov.nat HECTYPE 4 4 i
additem streamcov.nat streamcov.nat HECID 4 4 i
additem streamcov.nat streamcov.nat HECUPID 4 4 i
additem streamcov.nat streamcov.nat HECDNID 4 4 i
additem streamcov.nat streamcov.nat NODEZ 8 8 f 2
/*
/*--- streamcov.aat ---
/*
additem streamcov.aat streamcov.aat HECTYPE 4 4 i
additem streamcov.aat streamcov.aat HECID 4 4 i
additem streamcov.aat streamcov.aat HECUPID 4 4 i
additem streamcov.aat streamcov.aat HECDNID 4 4 i
/*
&if ( %attrib% = yes ) &then
&do
additem streamcov.aat streamcov.aat HECLENGTH 8 8 f 2
additem streamcov.aat streamcov.aat HECSLOPE 8 8 f 6
&end
/*
/*--- subcov.pat ---
/*
additem subcov.pat subcov.pat HECTYPE 4 4 i
additem subcov.pat subcov.pat HECID 4 4 i
additem subcov.pat subcov.pat HECDNID 4 4 i
/*
&if ( %attrib% = yes ) &then
&do
additem subcov.pat subcov.pat CENZ 8 8 f 2
additem subcov.pat subcov.pat MEANS 8 8 f 6
additem subcov.pat subcov.pat MEDIANS 8 8 f 6
additem subcov.pat subcov.pat B-COORD 8 8 c
additem subcov.pat subcov.pat F-COORD 8 8 c
additem subcov.pat subcov.pat FUPZ 8 8 f 2
additem subcov.pat subcov.pat FLENGTH 8 8 f 2
additem subcov.pat subcov.pat FLENGTH2 8 8 f 2
&end
```

```
/*
/*-----------------------------------------------
/*--- Mark Elements as Unknown Hectype (1.6.) ---
/*-----------------------------------------------
/*
&data arc arcplot
/*-ls0010
calculate streamcov.nat info HECTYPE = -1
calculate streamcov.aat info HECTYPE = -1
q
/*-le0010
&end
/*
/*-------------------------------------------------------------
/*-------------------------------------------------------------
/*--- Step 2. Establish Source, Sink and Subbasin Outlet ---
/*--- Elements -----------------------------------------------
/*-------------------------------------------------------------
/*-------------------------------------------------------------
/*
&type HECPREPRO:
&type HECPREPRO: Step 2. Establish Source, Sink and Subbasin Outlet
&type HECPREPRO: Elements
&type HECPREPRO:
&if ( %.oblevel% = .20 ) &then
&r hecpause 0
/*
/*----------------------------
/*--- Create Nodes (2.1.) ---
/*----------------------------
/*
&if [exists hydrocov -cover] &then
kill hydrocov all
intersect streamcov subcov hydrocov line # join
/*
/*-------------------------
/*--- Snap Nodes (2.2.) ---
/*-------------------------
/*
/*snap nodes in hydrocov to nodes in subcov.  this might be needed if
/*there are subbasin nodes at or near stream junctions.  since
/*this is most probably a result of delineating watersheds with grid
/*the %snaptol% variable should be set with the gridsize in mind.
/*three times the grid cell size is recommended.  process is optional.
/*
/*note that we have to identify which nodes eminated from a snap, because
/*they are subbasin outlets or sinks.  this is done with the snapitem.
/*simply taking the new nodes as subbasin outlets or sinks is not safe
/*because the snapped node might come from an intersection.
/*
&if ( %snapit% = yes ) &then
&do
/*-ls1005
/*
/*--- add snaped item to subcov.nat and hydrocov.nat ---
/*
additem subcov.nat subcov.nat SNAPED 4 4 i
build hydrocov node
additem hydrocov.nat hydrocov.nat SNAPED 4 4 i
/*
/*--- set subcov.pat snaped to 1 in arcplot ---
/*
&data arc arcplot
/*-ls1010
calculate subcov.nat info SNAPED = 1
q
/*-le1010
&end
/*
/*--- enter arcedit and put data to screen ---
/*
```

```
arcedit
edit hydrocov
editfeature arc
snapcoverage subcov
snapfeatures node node
&if ( %.oblevel% ne 0 ) &then
&do
&if [extract 1 [show display]] ne 9999 &then
display 9999
&end
mape subcov
&if ( %.oblevel% ne 0 ) &then
&do
drawenv arcs on
drawenv nodes on
backcov subcov
backenv arcs on
backenv nodes on
draw
&end
snapping closest %snaptol%
snapitems SNAPED
aselect all
snap
save
q
/*-le1005
&end
/*
/*-----------------------------------------------------
/*--- Set Up Hydrocov for Further Processing (2.3.) ---
/*-----------------------------------------------------
/*
/*--- rebuild topology (below might be too much) ---
/*
clean hydrocov hydrocov %snaptol% # line
clean hydrocov hydrocov %snaptol% # poly
build hydrocov node
build hydrocov arc
build hydrocov poly
/*
/*--- add xy values, because new nodes were added ---
/*
addxy hydrocov node
/*
/*-----------------------------------------------------
/*--- Calculate Surface Elevation Attributes (2.4.) ---
/*-----------------------------------------------------
/*
/*--- enter grid and put data to screen ---
/*
grid
&if ( %.oblevel% ne 0 ) &then
&do
&if [extract 1 [show display]] ne 9999 &then
display 9999
&end
/*
/*if user observation level is 4 we allow for change
/*in screen size to get a better look
/*
&if ( %.oblevel% = 4 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: Oportunity to change the display window size
&type HECPREPRO:
&r hecpause 4
&end
/*
&if ( %.oblevel% ne 0 ) &then
&do
```

```
mape %elevgrid%
gridpaint %elevgrid% value linear nowrap gray
&end
/*
/*--- subbasin attributes (2.4.1.) ---
/*
&if ( %attrib% = yes ) &then
&do
/*-ls1030
/*
/*calculate slope grid for the whole subbasin
/*
&if [exists slopegrid -grid] &then
kill slopegrid all
&type HECPREPRO:
&type HECPREPRO: Calculating slope
&type HECPREPRO:
slopegrid = ( slope (%elevgrid%,percentrise) / 100 )
&if ( %.oblevel% ne 0 ) &then
gridpaint slopegrid value linear nowrap gray
/*
&if [exists meansgrid -grid] &then
kill meansgrid all
&type HECPREPRO:
&type HECPREPRO: Calculating subbasin mean slopes
&type HECPREPRO:
meansgrid = zonalmean (subgrid,slopegrid,#)
&if ( %.oblevel% ne 0 ) &then
gridpaint meansgrid
/*
&if [exists mediansgrid -grid] &then
kill mediansgrid all
&type HECPREPRO:
&type HECPREPRO: Calculating subbasin median slopes
&type HECPREPRO: (please be patient, this might take a while)
&type HECPREPRO:
/*
/*note that mediansgrid is multiplied by 1000 in order to get
/*better accuracy.  later medians is divided again by 1000.
/*
mediansgrid = zonalmedian (subgrid, INT (slopegrid * 1000), #)
&if ( %.oblevel% ne 0 ) &then
&do
gridpaint mediansgrid
markersymbol 7
points subcov
textcolor 3
pointtext subcov subcov#
&end
/*
cursor cur1020 declare subcov.pat info rw
cursor cur1020 open
/*
/*skip first record, the universal polygon
/*
cursor cur1020 next
/*
/*loop once per subbasin
/*
&sv eof1020 = .FALSE.
&do &until %eof1020%
/*-ls1040
/*
/*get attributes
/*
&sv cenz = [show cellvalue %elevgrid% %:cur1020.x-coord% %:cur1020.y-coord%]
&sv means = [show cellvalue meansgrid %:cur1020.x-coord% %:cur1020.y-coord%]
&sv medians = [show cellvalue mediansgrid %:cur1020.x-coord% %:cur1020.y-
coord%]
&sv medians = %medians% / 1000
/*
```

```
/*write attributes to pat
/*
&s :cur1020.cenz = %cenz%
&s :cur1020.means = %means%
&s :cur1020.medians = %medians%
&type HECPREPRO:
&type HECPREPRO: Recording CENZ, MEANS and MEDIANS for subbasin
&type HECPREPRO: POLY: %:cur1020.subcov#%
&type HECPREPRO: CENZ: %cenz%
&type HECPREPRO: MEANS: %means%
&type HECPREPRO: MEDIANS: %medians%
&type HECPREPRO:
&r hecpause 3
/*
cursor cur1020 next
&if %:cur1020.AML$NEXT% = .FALSE. &then
   &sv eof1020 = .TRUE.
/*-le1040
&end
/*
cursor cur1020 remove
/*
kill slopegrid all
kill meansgrid all
kill mediansgrid all
kill subgrid all
/*
/*-le1030
&end
/*
/*--- stream node attributes (2.4.2.) ---
/*
/*display features
/*
&if ( %.oblevel% ne 0 ) &then
&do
markersymbol 20
nodes hydrocov
textcolor 4
nodetext hydrocov hydrocov#
&end
/*
cursor cur1010 declare hydrocov.nat info rw
cursor cur1010 open
/*
/*loop once per node
/*
&sv eof1010 = .FALSE.
&do &until %eof1010%
/*-ls1020
/*
/*get elevation from %elevgrid%
/*
&sv nodez = [show cellvalue %elevgrid% %:cur1010.x-coord% %:cur1010.y-coord%]
/*
/*write elevation to nat
/*
&s :cur1010.nodez = %nodez%
&type HECPREPRO:
&type HECPREPRO: Recording NODEZ for node
&type HECPREPRO: NODE: %:cur1010.hydrocov#%
&type HECPREPRO: NODEZ: %nodez%
&type HECPREPRO:
&r hecpause 3
/*
/*set up arrays
/*
&sv r#nodes%:cur1010.hydrocov#% = 0
&sv r#fnodes%:cur1010.hydrocov#% = 0
&sv r#tnodes%:cur1010.hydrocov#% = 0
&sv r#rnodes%:cur1010.hydrocov#% = 0
```

```
&sv r#rfnodes%:cur1010.hydrocov#% = 0
&sv r#rtnodes%:cur1010.hydrocov#% = 0
&sv rnodetype%:cur1010.hydrocov#% = unknown
/*
&sv h#nodes%:cur1010.hydrocov#% = 0
&sv h#fnodes%:cur1010.hydrocov#% = 0
&sv h#tnodes%:cur1010.hydrocov#% = 0
&sv h#rnodes%:cur1010.hydrocov#% = 0
&sv h#rfnodes%:cur1010.hydrocov#% = 0
&sv h#rtnodes%:cur1010.hydrocov#% = 0
&sv hnodetype%:cur1010.hydrocov#% = unknown
/*
&sv nhectype%:cur1010.hydrocov#% = 0
/*
/*&if ( %:cur1010.hectype% = 1 ) &then
/* &sv nhectype%:cur1010.hydrocov#% = 1
&sv nhecid%:cur1010.hydrocov#% = 0
/*is below needed?
&sv junrun%:cur1010.hydrocov#% = 0
/*
&sv nodez%:cur1010.hydrocov#% = %nodez%
/*
cursor cur1010 next
&if %:cur1010.AML$NEXT% = .FALSE. &then
   &sv eof1010 = .TRUE.
/*-le1020
&end
/*
cursor cur1010 remove
/*
/*leave grid
&if ( %.oblevel% ne 0 ) &then
display 0
q
/*
/*-------------------------------------------------------------
/*--- Establish Sources, Sinks and Subbasin Outlets (2.5.) ---
/*-------------------------------------------------------------
/*
/*--- enter arcplot and put data to screen ---
/*
arcplot
&if ( %.oblevel% ne 0 ) &then
&do
&if [extract 1 [show display]] ne 9999 &then
display 9999
&end
/*
/*if user observation level is 4 we allow for change
/*in screen size to get a better look
/*
&if ( %.oblevel% = 4 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: Oportunity to change the display window size
&type HECPREPRO:
&r hecpause 4
&end
/*
mape subcov
&if ( %.oblevel% ne 0 ) &then
&do
arcs streamcov
nodes streamcov
arcs subcov
&end
/*
/*--- 'clipped' procedure (2.5.1.) ---
/*
/*if the stream coverage allready had nodes at the intersection
/*with the subbasin coverage we will not be able to identify those
```

```
/*nodes based on them being created by the intersect command.  we have
/*go down all the nodes and see if there are actualy arcs from subcov
/*where the nodes are. this is most likely the case when the subbasin
/*coverage was used to cut out part of the stream coverage. optional.
/*
&if ( %clipped% = yes ) &then
&do
/*-ls1030
coordinate keyboard xy
searchtolerance 10
/*
cursor cur1030 declare hydrocov node rw
cursor cur1030 open
&sv eof1030 = .FALSE.
&do &until %eof1030%
/*-ls1040
unselect subcov arcs
aselect subcov arcs one *
%:cur1030.x-coord%,%:cur1030.y-coord%
&if ( [ extract 1 [ show select subcov line ] ] = 1 ) &then
&do
/*-ls1045
&s :cur1030.hectype = 0
/*-le1045
&end
cursor cur1030 next
&if %:cur1030.AML$NEXT% = .FALSE. &then
   &sv eof1030 = .TRUE.
/*-le1040
&end
cursor cur1030 remove
/*-le1030
&end
/*
aselect subcov arcs
/*
/*--- record data and create selection file (2.5.2.) ----
/*
unselect hydrocov node
aselect hydrocov node ( HECTYPE = 0 )
&if ( %snapit% = yes ) &then
&do
aselect hydrocov node ( SNAPED = 1 )
&end
writeselect file1
&if ( %.oblevel% ne 0 ) &then
&do
markersymbol 13
&type HECPREPRO:
&type HECPREPRO: Marking subbasin outlets, sinks and sources
&type HECPREPRO: (white triangle)
&type HECPREPRO:
&r hecpause 3
nodes hydrocov
&end
/*
/*--- write HECTYPE to NAT ----
/*
calculate hydrocov node HECTYPE = 1
/*
/*----------------------------------------
/*----------------------------------------
/*--- Step 3. Establish Channel System ---
/*----------------------------------------
/*----------------------------------------
/*
&type HECPREPRO:
&type HECPREPRO: Step 3. Establish Channel System
&type HECPREPRO:
&if ( %.oblevel% = .20 ) &then
&r hecpause 0
```

```
/*
/*---------------------------------------
/*--- Trace Downstream of Outlet Nodes ---
/*---------------------------------------
/*
unselect hydrocov node
unselect hydrocov arc
aselect hydrocov node
aselect hydrocov arc
trace downstream hydrocov file2 # file1
&sv a = [delete file1 -file]
/*
/*-----------------------------------------------------
/*--- Mark Arcs and Nodes Downstream of Outlet Nodes ---
/*-----------------------------------------------------
/*
unselect hydrocov node
unselect hydrocov arc
readselect file2
&sv a = [delete file2 -file]
unselect hydrocov node ( HECTYPE = 1 )
&if ( %.oblevel% ne 0 ) &then
&do
linecolor 3
&type HECPREPRO:
&type HECPREPRO: Marking channel system (green)
&type HECPREPRO:
&r hecpause 3
arcs hydrocov
&end
calculate hydrocov node HECTYPE = 4
calculate hydrocov arc HECTYPE = 4
/*
/*-------------------------------------------------
/*--- Mark Arcs and Nodes Upstream of Outlet Nodes ---
/*-------------------------------------------------
/*
nselect hydrocov node
nselect hydrocov arc
unselect hydrocov node ( HECTYPE = 1 )
&if ( %.oblevel% ne 0 ) &then
&do
linecolor 2
&type HECPREPRO:
&type HECPREPRO: Marking non-channel system (red)
&type HECPREPRO:
&r hecpause 3
arcs hydrocov
&end
calculate hydrocov node HECTYPE = 2
calculate hydrocov arc HECTYPE = 2
/*
/*-----------------------
/*--- Mark Reservoirs ---
/*-----------------------
/*
unselect hydrocov poly
aselect hydrocov poly
unselect hydrocov poly ( HYDROCOV# = 1 )
&if ( %.oblevel% ne 0 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: Marking reservoirs (cyan)
&type HECPREPRO:
&r hecpause 3
linecolor 5
polygons hydrocov
&end
/*
/*-----------------------------------------------
/*-----------------------------------------------
```

```
/*--- Step 4. Establish Channel Elements --------
/*-----------------------------------------------
/*-----------------------------------------------
/*
&type HECPREPRO:
&type HECPREPRO: Step 4. Establish Channel Elements
&type HECPREPRO:
&if ( %.oblevel% = .30 ) &then
&r hecpause 0
/*
/*
/*Possible node types are:
/*unknown -
/*none -
/*updangling -
/*dndangling -
/*interior -
/*junction -
/*diversion -
/*mreservoir -
/*upreservoir -
/*dnreservoir -
/*dreservoir -
/*jreservoir -
/*
/*Note that each node has two node types which are needed.  The r-node
/*type takes into account all arcs whereas the h-node type only takes
/*into account arcs from the channel system delineated in STEP 20.
/*
/*-----------------------------------------
/*--- Show Node and Arc Numbers on Screen ---
/*-----------------------------------------
/*
/*this will only be done for user observation level 4
/*
&sv moretext = no
&if ( %.oblevel% = 4 ) &then
&do
/*-ls3010
unselect hydrocov nodes
unselect hydrocov arcs
aselect hydrocov nodes
aselect hydrocov arcs
&type HECPREPRO:
&type HECPREPRO: Labeling nodes (white) and arcs (cyan)
&type HECPREPRO:
&r hecpause 4
textcolor 1
nodetext hydrocov hydrocov#
textcolor 5
arctext hydrocov hydrocov#
/*
/*remember that this has been done for later redisplay
/*
&sv moretext = yes
/*-le3010
&end
/*
/*-----------------------------------------
/*--- Create Hydrotopology Arrays (4.1.) ---
/*-----------------------------------------
/*
/*here we also find out if there are any reservoirs
/*
&sv reservoirs = no
/*
cursor cur3020 declare hydrocov.aat info ro
cursor cur3020 open
&sv eof3020 = .FALSE.
&do &until %eof3020%
/*-ls3030
```

```
/*
/*--- r-node information ---
/*
&type HECPREPRO:
&type HECPREPRO: Processing arc %:cur3020.hydrocov#% ...
&type HECPREPRO:
&type HECPREPRO: ... for r-node information
&type HECPREPRO:
&r hecpause 4
/*
/*process fnode
/*
&sv r#nodes = [ value r#nodes%:cur3020.fnode#% ]
&sv r#fnodes = [ value r#fnodes%:cur3020.fnode#% ]
&sv r#tnodes = [ value r#tnodes%:cur3020.fnode#% ]
&sv r#rnodes = [ value r#rnodes%:cur3020.fnode#% ]
&sv r#rfnodes = [ value r#rfnodes%:cur3020.fnode#% ]
&sv r#rtnodes = [ value r#rtnodes%:cur3020.fnode#% ]
/*
&sv r#nodes%:cur3020.fnode#% = %r#nodes% + 1
&sv r#fnodes%:cur3020.fnode#% = %r#fnodes% + 1
/*
&if ( ( %:cur3020.rpoly#% ne 1 ) or ( %:cur3020.lpoly#% ne 1 ) ) &then
&do
   &sv r#rnodes%:cur3020.fnode#% = %r#rnodes% + 1
   &sv r#rfnodes%:cur3020.fnode#% = %r#rfnodes% + 1
&end
/*
/*process tnode
/*
&sv r#nodes = [ value r#nodes%:cur3020.tnode#% ]
&sv r#fnodes = [ value r#fnodes%:cur3020.tnode#% ]
&sv r#tnodes = [ value r#tnodes%:cur3020.tnode#% ]
&sv r#rnodes = [ value r#rnodes%:cur3020.tnode#% ]
&sv r#rfnodes = [ value r#rfnodes%:cur3020.tnode#% ]
&sv r#rtnodes = [ value r#rtnodes%:cur3020.tnode#% ]
/*
&sv r#nodes%:cur3020.tnode#% = %r#nodes% + 1
&sv r#tnodes%:cur3020.tnode#% = %r#tnodes% + 1
/*
&if ( ( %:cur3020.rpoly#% ne 1 ) or ( %:cur3020.lpoly#% ne 1 ) ) &then
&do
   &sv r#rnodes%:cur3020.tnode#% = %r#rnodes% + 1
   &sv r#rtnodes%:cur3020.tnode#% = %r#rtnodes% + 1
&end
/*
/*--- h-node information ---
/*
&if ( ( %:cur3020.hectype% = 1 ) or ( %:cur3020.hectype% = 4 ) ) &then
&do
/*-ls3060
/*
&type HECPREPRO: ... for h-node information
&type HECPREPRO:
&r hecpause 4
/*
/*process fnode
/*
&sv h#nodes = [ value h#nodes%:cur3020.fnode#% ]
&sv h#fnodes = [ value h#fnodes%:cur3020.fnode#% ]
&sv h#tnodes = [ value h#tnodes%:cur3020.fnode#% ]
&sv h#rnodes = [ value h#rnodes%:cur3020.fnode#% ]
&sv h#rfnodes = [ value h#rfnodes%:cur3020.fnode#% ]
&sv h#rtnodes = [ value h#rtnodes%:cur3020.fnode#% ]
/*
&sv h#nodes%:cur3020.fnode#% = %h#nodes% + 1
&sv h#fnodes%:cur3020.fnode#% = %h#fnodes% + 1
/*
&if ( ( %:cur3020.rpoly#% ne 1 ) or ( %:cur3020.lpoly#% ne 1 ) ) &then
&do
   &sv h#rnodes%:cur3020.fnode#% = %h#rnodes% + 1
```

```
   &sv h#rfnodes%:cur3020.fnode#% = %h#rfnodes% + 1
   &sv reservoirs = yes
&end
/*
/*process tnode
/*
&sv h#nodes = [ value h#nodes%:cur3020.tnode#% ]
&sv h#fnodes = [ value h#fnodes%:cur3020.tnode#% ]
&sv h#tnodes = [ value h#tnodes%:cur3020.tnode#% ]
&sv h#rnodes = [ value h#rnodes%:cur3020.tnode#% ]
&sv h#rfnodes = [ value h#rfnodes%:cur3020.tnode#% ]
&sv h#rtnodes = [ value h#rtnodes%:cur3020.tnode#% ]
/*
&sv h#nodes%:cur3020.tnode#% = %h#nodes% + 1
&sv h#tnodes%:cur3020.tnode#% = %h#tnodes% + 1
/*
&if ( ( %:cur3020.rpoly#% ne 1 ) or ( %:cur3020.lpoly#% ne 1 ) ) &then
&do
   &sv h#rnodes%:cur3020.tnode#% = %h#rnodes% + 1
   &sv h#rtnodes%:cur3020.tnode#% = %h#rtnodes% + 1
&end
/*
/*-le3060
&end
/*
cursor cur3020 next
&if %:cur3020.AML$NEXT% = .FALSE. &then
   &sv eof3020 = .TRUE.
/*-le3030
&end
cursor cur3020 remove
/*
/*select complete channel network
/*
aselect hydrocov node
aselect hydrocov arc
/*
/*-----------------------------------
/*--- Calculate Node Types (4.2.) ---
/*-----------------------------------
/*
/*--- set up hecid variable ---
/*
&sv hecid = 1
/*
cursor cur3030 declare hydrocov.nat info rw
cursor cur3030 open
&sv eof3030 = .FALSE.
&do &until %eof3030%
/*-ls3040
/*
/*--- pull values from memory ---
/*
&sv r#nodes = [ value r#nodes%:cur3030.hydrocov#% ]
&sv r#fnodes = [ value r#fnodes%:cur3030.hydrocov#% ]
&sv r#tnodes = [ value r#tnodes%:cur3030.hydrocov#% ]
&sv r#rnodes = [ value r#rnodes%:cur3030.hydrocov#% ]
&sv r#rfnodes = [ value r#rfnodes%:cur3030.hydrocov#% ]
&sv r#rtnodes = [ value r#rtnodes%:cur3030.hydrocov#% ]
&sv rnodetype = [ value rnodetype%:cur3030.hydrocov#% ]
/*
&sv h#nodes = [ value h#nodes%:cur3030.hydrocov#% ]
&sv h#fnodes = [ value h#fnodes%:cur3030.hydrocov#% ]
&sv h#tnodes = [ value h#tnodes%:cur3030.hydrocov#% ]
&sv h#rnodes = [ value h#rnodes%:cur3030.hydrocov#% ]
&sv h#rfnodes = [ value h#rfnodes%:cur3030.hydrocov#% ]
&sv h#rtnodes = [ value h#rtnodes%:cur3030.hydrocov#% ]
&sv hnodetype = [ value hnodetype%:cur3030.hydrocov#% ]
/*
/*--- r-node type ---
/*
```

```
/*updangling
/*
&if ( ( %r#nodes% = 1 ) and ( %r#fnodes% = 1 ) ) &then
&sv rnodetype = updangling
/*
/*dndangling
/*
&if ( ( %r#nodes% = 1 ) and ( %r#tnodes% = 1 ) ) &then
&sv rnodetype = dndangling
/*
/*interior
/*
&if ( ( %r#fnodes% = 1 ) and ( %r#tnodes% = 1 ) ) &then
&sv rnodetype = interior
/*
/*junction
/*
&if ( ( %r#fnodes% = 1 ) and ( %r#tnodes% >= 2 ) ) &then
&sv rnodetype = junction
/*
/*diversion
/*
&if ( ( %r#tnodes% = 1 ) and ( %r#fnodes% >= 2 ) ) &then
&sv rnodetype = diversion
/*
/*mreservoir
/*
&if ( ( %r#rfnodes% = 1 ) and ( %r#rtnodes% = 1 ) ) &then
&sv rnodetype = mreservoir
/*
/*upreservoir
/*
&if ( ( %r#rfnodes% = 2 ) and ( %r#rtnodes% = 0 ) ) &then
&sv rnodetype = upreservoir
/*
/*dnreservoir
/*
&if ( ( %r#rtnodes% = 2 ) and ( %r#rfnodes% = 0 ) ) &then
&sv rnodetype = dnreservoir
/*
/*dreservoir
/*
&if ( ( %r#rtnodes% = 1 ) and ( %r#rfnodes% = 1 ) and ( %r#fnodes% = 2 ) )
&then
&sv rnodetype = dreservoir
/*
/*jreservoir
/*
&if ( ( %r#rtnodes% = 1 ) and ( %r#rfnodes% = 1 ) and ( %r#tnodes% = 2 ) )
&then
&sv rnodetype = jreservoir
/*
/*write to array
/*
&sv rnodetype%:cur3030.hydrocov#% = %rnodetype%
/*
/*--- h-node type ---
/*
/*updangling
/*
&if ( ( %h#nodes% = 1 ) and ( %h#fnodes% = 1 ) ) &then
&sv hnodetype = updangling
/*
/*dndangling
/*
&if ( ( %h#nodes% = 1 ) and ( %h#tnodes% = 1 ) ) &then
&sv hnodetype = dndangling
/*
/*interior
/*
&if ( ( %h#fnodes% = 1 ) and ( %h#tnodes% = 1 ) ) &then
```

```
&sv hnodetype = interior
/*
/*junction
/*
&if ( ( %h#fnodes% = 1 ) and ( %h#tnodes% >= 2 ) ) &then
&sv hnodetype = junction
/*
/*diversion
/*
&if ( ( %h#tnodes% = 1 ) and ( %h#fnodes% >= 2 ) ) &then
&sv hnodetype = diversion
/*
/*mreservoir
/*
&if ( ( %h#rfnodes% = 1 ) and ( %h#rtnodes% = 1 ) ) &then
&sv hnodetype = mreservoir
/*
/*upreservoir
/*
&if ( ( %h#rfnodes% = 2 ) and ( %h#rtnodes% = 0 ) ) &then
&sv hnodetype = upreservoir
/*
/*dnreservoir
/*
&if ( ( %h#rtnodes% = 2 ) and ( %h#rfnodes% = 0 ) ) &then
&sv hnodetype = dnreservoir
/*
/*dreservoir
/*
&if ( ( %h#rtnodes% = 1 ) and ( %h#rfnodes% = 1 ) and ( %h#fnodes% = 2 ) )
&then
&sv hnodetype = dreservoir
/*
/*jreservoir
/*
&if ( ( %h#rtnodes% = 1 ) and ( %h#rfnodes% = 1 ) and ( %h#tnodes% = 2 ) )
&then
&sv hnodetype = jreservoir
/*
/*none
/*
&if ( ( %h#nodes% = 0 ) and ( %h#rnodes% = 0 ) ) &then
&sv hnodetype = none
/*
/*write to array
/*
&sv hnodetype%:cur3030.hydrocov#% = %hnodetype%
/*
/*--- screen output and error check ---
/*
&type HECPREPRO:
&type HECPREPRO: Node information
&type HECPREPRO:
&type HECPREPRO: NODE: %:cur3030.hydrocov#%
&type HECPREPRO:
&type HECPREPRO: R#NODES: %r#nodes%
&type HECPREPRO: R#FNODES: %r#fnodes%
&type HECPREPRO: R#TNODES: %r#tnodes%
&type HECPREPRO: R#RNODES: %r#rnodes%
&type HECPREPRO: R#RFNODES: %r#rfnodes%
&type HECPREPRO: R#RTNODES: %r#rtnodes%
&type HECPREPRO: RNODETYPE: %rnodetype%
&type HECPREPRO:
&type HECPREPRO: H#NODES: %h#nodes%
&type HECPREPRO: H#FNODES: %h#fnodes%
&type HECPREPRO: H#TNODES: %h#tnodes%
&type HECPREPRO: H#RNODES: %h#rnodes%
&type HECPREPRO: H#RFNODES: %h#rfnodes%
&type HECPREPRO: H#RTNODES: %h#rtnodes%
&type HECPREPRO: HNODETYPE: %hnodetype%
&type HECPREPRO:
```

```
&r hecpause 4
/*
/*if the rnodetype is unknown we have an error condition
/*
&if ( %rnodetype% = unknown ) &then
&do
&type HECPREPRO:
&type HECPREPRO: ERROR: UNIDENTIFIABLE RNODETYPE
&type HECPREPRO:
&r hecpause 1
&sv errors = %errors% + 1
&end
/*
/*if the hnodetype is unknown we have an error condition
&type interior = %hnodetype%
&if ( %hnodetype% = unknown ) &then
&do
&type HECPREPRO:
&type HECPREPRO: ERROR: UNIDENTIFIABLE HNODETYPE
&type HECPREPRO:
&r hecpause 1
&sv errors = %errors% + 1
&end
/*
/*---------------------------------------------------------
/*--- Classify Elements Based on Hectype System (4.3) ---
/*---------------------------------------------------------
/*
/*--- junctions ---
/*
/*note that there is a possibility that a subbasin outlet is also
/*a junction.  in that case we will keep it's hectype as 1, because
/*we need it later for subbasin to subbasin outlet connectivity.
/*
&if ( ( %hnodetype% = junction ) and ( %:cur3030.hectype% ne 1 ) ) &then
&do
&s :cur3030.hectype = 3
&s :cur3030.hecid = %hecid%
&type HECPREPRO: Recording HECID, HECTYPE, HECX and HECY
&type HECPREPRO:
&type HECPREPRO: JUNCTION
&type HECPREPRO: HECID: %hecid%
&type HECPREPRO: HECTYPE: %:cur3030.hectype%
&type HECPREPRO: HECX: %:cur3030.x-coord%
&type HECPREPRO: HECY: %:cur3030.y-coord%
&type HECPREPRO:
&r hecpause 3
/*
&sv hectype%hecid% = %:cur3030.hectype%
&sv hecx%hecid% = %:cur3030.x-coord%
&sv hecy%hecid% = %:cur3030.y-coord%
&sv hh#fnodes%hecid% = [ value h#fnodes%:cur3030.hydrocov#% ]
&sv hh#tnodes%hecid% = [ value h#tnodes%:cur3030.hydrocov#% ]
/*
&sv nhectype%:cur3030.hydrocov#% = %:cur3030.hectype%
&sv nhecid%:cur3030.hydrocov#% = %hecid%
/*
&sv hecid = %hecid% + 1
/*
&end
/*
/*--- diversions ---
/*
&if ( %hnodetype% = diversion ) &then
&do
&s :cur3030.hectype = 8
&s :cur3030.hecid = %hecid%
&type HECPREPRO: Recording HECID, HECTYPE, HECX and HECY
&type HECPREPRO:
&type HECPREPRO: DIVERSION
&type HECPREPRO: HECID: %hecid%
```

```
&type HECPREPRO: HECTYPE: %:cur3030.hectype%
&type HECPREPRO: HECX: %:cur3030.x-coord%
&type HECPREPRO: HECY: %:cur3030.y-coord%
&type HECPREPRO:
&r hecpause 3
/*
&sv hectype%hecid% = %:cur3030.hectype%
&sv hecx%hecid% = %:cur3030.x-coord%
&sv hecy%hecid% = %:cur3030.y-coord%
&sv hh#fnodes%hecid% = [ value h#fnodes%:cur3030.hydrocov#% ]
&sv hh#tnodes%hecid% = [ value h#tnodes%:cur3030.hydrocov#% ]
/*
&sv nhectype%:cur3030.hydrocov#% = %:cur3030.hectype%
&sv nhecid%:cur3030.hydrocov#% = %hecid%
/*
&sv hecid = %hecid% + 1
/*
&end
/*
/*--- sinks ---
/*
&if ( %rnodetype% = dndangling ) &then
&do
&s :cur3030.hectype = 6
&s :cur3030.hecid = %hecid%
&type HECPREPRO: Recording HECID, HECTYPE, HECX and HECY
&type HECPREPRO:
&type HECPREPRO: SINK
&type HECPREPRO: HECID: %hecid%
&type HECPREPRO: HECTYPE: %:cur3030.hectype%
&type HECPREPRO: HECX: %:cur3030.x-coord%
&type HECPREPRO: HECY: %:cur3030.y-coord%
&type HECPREPRO:
&r hecpause 3
/*
&sv hectype%hecid% = %:cur3030.hectype%
&sv hecx%hecid% = %:cur3030.x-coord%
&sv hecy%hecid% = %:cur3030.y-coord%
&sv hh#fnodes%hecid% = [ value h#fnodes%:cur3030.hydrocov#% ]
&sv hh#tnodes%hecid% = [ value h#tnodes%:cur3030.hydrocov#% ]
/*
&sv nhectype%:cur3030.hydrocov#% = %:cur3030.hectype%
&sv nhecid%:cur3030.hydrocov#% = %hecid%
/*
&sv hecid = %hecid% + 1
/*
&end
/*
/*--- sources ---
/*
&if ( ( %rnodetype% = updangling ) and ( %hnodetype% = updangling ) ) &then
&do
&s :cur3030.hectype = 7
&s :cur3030.hecid = %hecid%
&type HECPREPRO: Recording HECID, HECTYPE, HECX and HECY
&type HECPREPRO:
&type HECPREPRO: SOURCE
&type HECPREPRO: HECID: %hecid%
&type HECPREPRO: HECTYPE: %:cur3030.hectype%
&type HECPREPRO: HECX: %:cur3030.x-coord%
&type HECPREPRO: HECY: %:cur3030.y-coord%
&type HECPREPRO:
&r hecpause 3
/*
&sv hectype%hecid% = %:cur3030.hectype%
&sv hecx%hecid% = %:cur3030.x-coord%
&sv hecy%hecid% = %:cur3030.y-coord%
&sv hh#fnodes%hecid% = [ value h#fnodes%:cur3030.hydrocov#% ]
&sv hh#tnodes%hecid% = [ value h#tnodes%:cur3030.hydrocov#% ]
/*
&sv nhectype%:cur3030.hydrocov#% = %:cur3030.hectype%
```

```
&sv nhecid%:cur3030.hydrocov#% = %hecid%
/*
&sv hecid = %hecid% + 1
/*
&end
/*
/*--- reservoir ---
/*
&if ( %hnodetype% = dnreservoir ) &then
&do
&s :cur3030.hectype = 10
&s :cur3030.hecid = %hecid%
&type HECPREPRO: Recording HECID, HECTYPE, HECX and HECY
&type HECPREPRO:
&type HECPREPRO: RESERVOIR
&type HECPREPRO: HECID: %hecid%
&type HECPREPRO: HECTYPE: %:cur3030.hectype%
&type HECPREPRO: HECX: %:cur3030.x-coord%
&type HECPREPRO: HECY: %:cur3030.y-coord%
&type HECPREPRO:
&r hecpause 3
/*
&sv hectype%hecid% = %:cur3030.hectype%
&sv hecx%hecid% = %:cur3030.x-coord%
&sv hecy%hecid% = %:cur3030.y-coord%
&sv hh#fnodes%hecid% = [ value h#fnodes%:cur3030.hydrocov#% ]
/*
&sv nhectype%:cur3030.hydrocov#% = %:cur3030.hectype%
&sv nhecid%:cur3030.hydrocov#% = %hecid%
/*
/*below is special for reservoirs and has to do with
/*multiple hecupid identification
&sv h#tnodes = [ value h#tnodes%:cur3030.hydrocov#% ]
&sv h#rtnodes = [ value h#rtnodes%:cur3030.hydrocov#% ]
&sv h#tnodes = ( %h#tnodes% - %h#rtnodes% )
&sv h#tnodes%:cur3030.hydrocov#% = %h#tnodes%
&sv hh#tnodes%hecid% = %h#tnodes%
/*
&sv hecid = %hecid% + 1
/*
&end
/*
/*--- subbasin outlet ---
/*
&if ( %:cur3030.hectype% = 1 ) &then
&do
&s :cur3030.hecid = %hecid%
&type HECPREPRO: Recording HECID, HECTYPE, HECX and HECY
&type HECPREPRO:
&type HECPREPRO: SUBBASIN OUTLET
&type HECPREPRO: HECID: %hecid%
&type HECPREPRO: HECTYPE: %:cur3030.hectype%
&type HECPREPRO: HECX: %:cur3030.x-coord%
&type HECPREPRO: HECY: %:cur3030.y-coord%
&type HECPREPRO:
&r hecpause 3
/*
&sv hectype%hecid% = %:cur3030.hectype%
&sv hecx%hecid% = %:cur3030.x-coord%
&sv hecy%hecid% = %:cur3030.y-coord%
&sv hh#fnodes%hecid% = [ value h#fnodes%:cur3030.hydrocov#% ]
&sv hh#tnodes%hecid% = [ value h#tnodes%:cur3030.hydrocov#% ]
/*
&sv nhectype%:cur3030.hydrocov#% = %:cur3030.hectype%
&sv nhecid%:cur3030.hydrocov#% = %hecid%
/*
&sv hecid = %hecid% + 1
/*
&end
/*
/*close loop
```

```
/*
cursor cur3030 next
&if %:cur3030.AML$NEXT% = .FALSE. &then
   &sv eof3030 = .TRUE.
/*-le3040
&end
cursor cur3030 remove
/*
/*-----------------------------------------------------------
/*--- Calculate Subbasin Outlet Node Information (4.4.) ---
/*-----------------------------------------------------------
/*
cursor cur3040 declare hydrocov.aat info ro
cursor cur3040 open
&sv eof3040 = .FALSE.
&do &until %eof3040%
/*-ls3050
/*
&type HECPREPRO:
&type HECPREPRO: Processing for subbasin information
&type HECPREPRO:
&type HECPREPRO: ARC: %:cur3040.hydrocov#%
&type HECPREPRO:
&r hecpause 4
/*
&sv outlet = no
&sv hectype = [ value nhectype%:cur3040.tnode#% ]
&if ( %hectype% = 1 ) &then
   &sv outlet = yes
&if ( %hectype% = 6 ) &then
   &sv outlet = yes
/*is below really needed?
/*-X
&sv rnodetype = [ value rnodetype%:cur3040.tnode#% ]
&if ( %rnodetype% = updangling ) &then
   &sv outlet = no
&sv hnodetype = [ value hnodetype%:cur3040.tnode#% ]
/*-X
/*
&if ( %outlet% = yes ) &then
&do
/*-ls3060
/*
/*if this is not the first outlet we encounter we have to check
/*the eleevation of the outlet, because only the lowest elevation
/*is the subbasin outlet.  multiple outlets can occur if there is a
/*diversion in the subbasin.
&sv multipleoutlets = no
&sv polydnnode = [ value polydnnode%:cur3040.subcov#% ]
&if ( [ length %polydnnode% ] ne 0 ) &then
&do
/*-ls3070
&sv multipleoutlets = yes
&sv currentnode = %:cur3040.tnode#%
&sv currentz = [value nodez%:cur3040.tnode#% ]
&sv previousnode = [ value polydnnode%:cur3040.subcov#% ]
&sv previousz = [ value nodez%previousnode% ]
&sv updateoutlet = no
&if ( %currentz% < %previousz% ) &then
   &sv updateoutlet = yes
&if ( %updateoutlet% = yes ) &then
   &sv polydnnode%:cur3040.subcov#% = %currentnode%
/*-le3070
&end
/*
&if ( [ length %polydnnode% ] = 0 ) &then
   &sv polydnnode%:cur3040.subcov#% = %:cur3040.tnode#%
/*
/*we also have to update the h#tnodes variables
&sv h#tnodes = [ value h#tnodes%:cur3040.tnode#% ]
&sv h#tnodes%:cur3040.tnode#% = ( %h#tnodes% + 1 )
```

```
/*
&type HECPREPRO:
&type HECPREPRO: Recording subbasin information
&type HECPREPRO:
&type HECPREPRO: ARC: %:cur3040.hydrocov#%
&type HECPREPRO: TNODE: %:cur3040.tnode#%
&type HECPREPRO: HECTYPE: %hectype%
&type HECPREPRO: RNODETYPE: %rnodetype%
&type HECPREPRO: HNODETYPE: %hnodetype%
&type HECPREPRO: POLY: %:cur3040.subcov#%
&type HECPREPRO:
&if ( %multipleoutlets% = yes ) &then
&do
&type HECPREPRO:
&type HECPREPRO: MULTIPLE OUTLET INFORMATION
&type HECPREPRO:
&type HECPREPRO: CURRENTNODE: %currentnode%
&type HECPREPRO: CURRENTZ: %currentz%
&type HECPREPRO:
&type HECPREPRO: PREVIOUSNODE: %previousnode%
&type HECPREPRO: PREVIOUSZ: %previousz%
&type HECPREPRO:
&type HECPREPRO: UPDATEOUTLET: %updateoutlet%
&type HECPREPRO:
&end
&r hecpause 3
/*-le3060
&end
/*
cursor cur3040 next
&if %:cur3040.AML$NEXT% = .FALSE. &then
   &sv eof3040 = .TRUE.
/*-le3050
&end
cursor cur3040 remove
/*
/*-----------------------------------------------
/*--- Clear Node and Arc Numbers from Screen ---
/*-----------------------------------------------
/*
/*this will only be done if they were displayed
&if ( %moretext% = yes ) &then
&do
/*-ls4020
&type HECPREPRO:
&type HECPREPRO: Clearing display
&type HECPREPRO:
&r hecpause 4
clear
/*base data
linecolor 1
markersymbol 17
arcs streamcov
nodes streamcov
arcs subcov
/*channel system
unselect hydrocov node
unselect hydrocov arc
aselect hydrocov node ( HECTYPE = 4 )
aselect hydrocov arc ( HECTYPE = 4 )
linecolor 3
arcs hydrocov
/*non-channel system
unselect hydrocov node
unselect hydrocov arc
aselect hydrocov node ( HECTYPE = 2 )
aselect hydrocov arc ( HECTYPE = 2 )
linecolor 2
arcs hydrocov
/*reservoirs
unselect hydrocov poly
```

```
aselect hydrocov poly
unselect hydrocov poly ( HYDROCOV# = 1 )
linecolor 5
polygons hydrocov
/*-le4020
&end
/*
/*----------------------
/*--- Mark Elements ---
/*----------------------
/*
&if ( %.oblevel% ne 0 ) &then
&do
/*-ls4030
/*
textcolor 1
/*
/*--- subbasin outlets ---
/*
&type HECPREPRO:
&type HECPREPRO: Marking subbasin outlets (green triangle)
&type HECPREPRO:
&r hecpause 3
unselect hydrocov node
aselect hydrocov node ( hectype = 1 )
markersymbol 15
nodes hydrocov
nodetext hydrocov hecid # LL
/*
/*--- junctions ---
/*
&type HECPREPRO:
&type HECPREPRO: Marking junctions (green circle)
&type HECPREPRO:
&r hecpause 3
unselect hydrocov node
aselect hydrocov node ( hectype = 3 )
markersymbol 11
nodes hydrocov
nodetext hydrocov hecid # LL
/*
/*--- diversions ---
/*
&type HECPREPRO:
&type HECPREPRO: Marking diversions (blue circle)
&type HECPREPRO:
&r hecpause 3
unselect hydrocov node
aselect hydrocov node ( hectype = 8 )
markersymbol 12
nodes hydrocov
nodetext hydrocov hecid # LL
/*
/*--- sinks ---
/*
&type HECPREPRO:
&type HECPREPRO: Marking sinks (red triangle)
&type HECPREPRO:
&r hecpause 3
unselect hydrocov node
aselect hydrocov node ( hectype = 6 )
markersymbol 14
nodes hydrocov
nodetext hydrocov hecid # LL
/*
/*--- sources ---
/*
&type HECPREPRO:
&type HECPREPRO: Marking sources (blue triangle)
&type HECPREPRO:
&r hecpause 3
```

```
unselect hydrocov node
aselect hydrocov node ( hectype = 7 )
markersymbol 16
nodes hydrocov
nodetext hydrocov hecid # LL
/*
/*--- reservoirs ---
/*
&type HECPREPRO:
&type HECPREPRO: Marking reservoirs (red upsidedown triangle)
&type HECPREPRO:
&r hecpause 3
unselect hydrocov node
aselect hydrocov node ( hectype = 10 )
markersymbol 42
nodes hydrocov
nodetext hydrocov hecid # LL
/*
/*-le4030
&end
/*
/*--------------------------------------
/*--------------------------------------
/*--- Step 5. Calculate Connectivity ---
/*--------------------------------------
/*--------------------------------------
/*
&type HECPREPRO:
&type HECPREPRO: Step 5. Calculate Connectivity
&type HECPREPRO:
&if ( %.oblevel% = .60 ) &then
&r hecpause 0
/*
/*--------------------------------------
/*-- Channel Elements and Reaches 5.1. --
/*--------------------------------------
/*
/*--- display current and processed reach arcs in yellow ---
/*
&if ( %.oblevel% ne 0 ) &then
linecolor 7
/*
/*--- proceed downstream from upstream nodes in channel system ---
/*
/*process all the nodes belonging to the channel system as
/*possible upstream ends of reaches
/*
unselect hydrocov.nat info ( hectype = 2 )
cursor cur6020 declare hydrocov.nat info rw
cursor cur6020 open
&sv eof6020 = .FALSE.
&sv divrun = 0
/*
&do &until %eof6020%
/*-ls6020
/*
/*--- determine if node is an upstream element (5.1.1.) ---
/*
/*check the nhectype of the node. the node is a valid element for this
/*operation if it is hectype 1, 3, 7, 8, 10 or if it is a reservoir diversion
/*(dreservoir). we also have to remember to go back to this node once more if
/*it is a diversion, hectype 8.
/*
&sv nhecid = [ value nhecid%:cur6020.hydrocov#% ]
&sv rnodetype = [ value rnodetype%:cur6020.hydrocov#% ]
&sv hnodetype = [ value hnodetype%:cur6020.hydrocov#% ]
&sv nhectype = [ value nhectype%:cur6020.hydrocov#% ]
&sv h#fnodes = [ value h#fnodes%:cur6020.hydrocov#% ]
&sv valid = no
&sv diversion = no
/*
```

```
&if ( %nhectype% = 1 ) &then
&sv valid = yes
/*
&if ( %nhectype% = 3 ) &then
&sv valid = yes
/*
&if ( %nhectype% = 7 ) &then
&sv valid = yes
/*
&if ( %nhectype% = 8 ) &then
&do
   &sv valid = yes
   &sv diversion = yes
   &sv divrun = %divrun% + 1
&end
/*
&if ( %nhectype% = 10 ) &then
&do
&sv valid = yes
&if ( %h#fnodes% >= 2 ) &then
   &do
   &sv diversion = yes
   &sv divrun = %divrun% + 1
   &end
&end
/*
&if ( %hnodetype% = dreservoir ) &then
&sv valid = yes
/*
&type HECPREPRO:
&type HECPREPRO: Searching for an upstream end of a reach
&type HECPREPRO:
&type HECPREPRO: ELEMENT INFORMATION
&type HECPREPRO:
&type HECPREPRO: FNODE: %:cur6020.hydrocov#%
&type HECPREPRO: HECID: %nhecid%
&type HECPREPRO:
&type HECPREPRO: RNODETYPE: %rnodetype%
&type HECPREPRO: HNODETYPE: %hnodetype%
&type HECPREPRO: HECTYPE: %nhectype%
&type HECPREPRO:
&type HECPREPRO: FNODE VALIDITY: %valid%
&type HECPREPRO:
&if ( %diversion% = yes ) &then
&do
&type HECPREPRO: DIVERSION INFORMATION
&type HECPREPRO: DIVRUN: %divrun%
&type HECPREPRO: H#FNODES: %h#fnodes%
&type HECPREPRO:
&end
&r hecpause 3
/*
/*proceed with the analysis only if the node is validated, otherwise
/*the next node is selected.
/*
&if ( %valid% = yes ) &then
&do
/*
/*clear the selection set
/*
unselect hydrocov arc
unselect hydrocov node
/*
/*--- trace downstream until downstream element is found (5.1.2.) ---
/*
/*find the one arc that has this node as fnode, which is the first downstream
/*arc.
/*
&sv fnode = %:cur6020.hydrocov#%
&if ( %attrib% = yes ) &then
&do
```

```
   &sv hecslope = 0
   &sv heclength = 0
&end
&sv arcs = ' '
&sv closed = .FALSE.
&do &until %closed%
/*-ls6030
/*
/*we only have to consider arcs from the channel system and we
/*will remove arcs from the channel system as we go so that we
/*don't process nunecessary arcs.
/*
unselect hydrocov.aat info
aselect hydrocov.aat info ( hectype = 4 )
unselect hydrocov.aat info ( hecid ne 0 )
/*
/*go down the AAT and look at all the arcs until suspect is found.
/*remember not to go to the same arc twice if we are at a diversion.
/*
cursor cur6030 declare hydrocov.aat info ro
cursor cur6030 open
&sv eof6030 = .FALSE.
&do &until %eof6030%
/*-ls6040
/*
&sv valid = no
&if ( %:cur6030.fnode#% = %fnode% ) &then
   &sv valid = yes
/*
/*
&if ( %valid% = yes ) &then
&do
/*-ls6045
/*
/*process connectivity variables
/*
&sv tnode = %:cur6030.tnode#%
&sv arc = %:cur6030.hydrocov#%
&sv arcs = %arcs% %:cur6030.hydrocov#%
&sv eof6030 = .TRUE.
/*
/*process attribute data
/*
&if ( %attrib% = yes ) &then
&do
/*-ls6047
&sv fnodez = [ value nodez%:cur6030.fnode#% ]
&sv tnodez = [ value nodez%:cur6030.tnode#% ]
&sv slope = ( ( %fnodez% - %tnodez% ) / %:cur6030.length% )
&sv drop = ( %hecslope% * %heclength% + %slope% * %:cur6030.length% )
&sv heclength = %heclength% + %:cur6030.length%
&sv hecslope = ( %drop% / %heclength% )
&type HECPREPRO:
&type HECPREPRO: REACH ATTRIBUTE INFORMATION
&type HECPREPRO:
&type HECPREPRO: HECLENGTH: %heclength%
&type HECPREPRO: HECSLOPE: %hecslope%
&type HECPREPRO:
&r hecpause 4
/*-le6047
&end
/*
/*-le6045
&end
/*
cursor cur6030 next
&if %:cur6030.AML$NEXT% = .FALSE. &then
   &sv eof6030 = .TRUE.
/*-le6040
&end
cursor cur6030 remove
```

```
/*
/*add the arc to the selection set and display it
/*
aselect hydrocov arcs ( hydrocov# = %arc% )
&if ( %.oblevel% ne 0 ) &then
arcs hydrocov
/*
/*if the arcs tnode is a valid element the end of the channel is found
/*and the reach is closed, otherwise the next downstream arc has to be
/*examined.  valid elements include HECTYPE 1, 3, 6, 8, 9 and reservoir
/*junctions (jreservoir).  note that with upreservoirs and jreservoirs
/*the channel closes, but we have to continue downstream along the shore
/*of the reservoir to find the dnreservoir.  note also that if a subbasin
/*outlet or a sink is found as a downstream end of a reach it is a junction
/*with multiple upstream elements.
/*how to handle sinks for this case is unclear.
/*
&sv nhecid = [ value nhecid%tnode% ]
&sv rnodetype = [ value rnodetype%tnode% ]
&sv hnodetype = [ value hnodetype%tnode% ]
&sv nhectype = [ value nhectype%tnode% ]
&sv h#tnodes = [ value h#tnodes%tnode% ]
&sv valid = no
&sv reservoir = no
&sv junction = no
/*
&if ( %nhectype% = 1 ) &then
&do
&sv valid = yes
&sv junction = yes
&sv junrun = [ value junrun%tnode% ]
&sv junrun = %junrun% + 1
&sv junrun%tnode% = %junrun%
&end
/*
&if ( %nhectype% = 3 ) &then
&do
&sv valid = yes
&sv junction = yes
&sv junrun = [ value junrun%tnode% ]
&sv junrun = %junrun% + 1
&sv junrun%tnode% = %junrun%
&end
/*
&if ( %nhectype% = 6 ) &then
&do
&sv valid = yes
&sv junction = yes
&sv junrun = [ value junrun%tnode% ]
&sv junrun = %junrun% + 1
&sv junrun%tnode% = %junrun%
&end
/*
&if ( %nhectype% = 8 ) &then
&sv valid = yes
/*
&if ( %nhectype% = 10 ) &then
&do
&sv valid = yes
&if ( %h#tnodes% > 1 ) &then
&do
&sv junction = yes
&sv junrun = [ value junrun%tnode% ]
&sv junrun = %junrun% + 1
&sv junrun%tnode% = %junrun%
&end
&end
/*
&if ( %hnodetype% = upreservoir ) &then
&do
   &sv valid = yes
```

```
   &sv reservoir = yes
&end
/*
&if ( %hnodetype% = jreservoir ) &then
&do
   &sv valid = yes
   &sv reservoir = yes
&end
/*
&type HECPREPRO:
&type HECPREPRO: Searching for downstream end of reach
&type HECPREPRO:
&type HECPREPRO: ELEMENT INFORMATION
&type HECPREPRO:
&type HECPREPRO: TNODE: %tnode%
&type HECPREPRO: HECID: %nhecid%
&type HECPREPRO:
&type HECPREPRO: RNODETYPE: %rnodetype%
&type HECPREPRO: HNODETYPE: %hnodetype%
&type HECPREPRO: HECTYPE: %nhectype%
&type HECPREPRO:
&type HECPREPRO: TNODE VALIDITY: %valid%
&type HECPREPRO:
&if ( %diversion% = yes ) &then
&do
&type HECPREPRO: DIVERSION INFORMATION
&type HECPREPRO: DIVRUN: %divrun%
&type HECPREPRO:
&end
&if ( %junction% = yes ) &then
&do
&type HECPREPRO: JUNCTION INFORMATION
&type HECPREPRO: JUNRUN: %junrun%
&type HECPREPRO:
&end
&if ( %reservoir% = yes ) &then
&do
&type HECPREPRO:
&type HECPREPRO: RESERVOIR
&type HECPREPRO:
/*
/*for the reservoir case the tnode becomes the new fnode
/*
&sv fnode = %tnode%
&end
/*
/*the reach closes if the node is valid
/*
&if ( %valid% = yes ) &then
&do
&type HECPREPRO:
&type HECPREPRO: Reach Closure
&type HECPREPRO:
&r hecpause 3
&sv closed = .TRUE.
&end
&if ( %valid% = no ) &then
&do
&type HECPREPRO:
&type HECPREPRO: No Reach Closure
&type HECPREPRO:
&r hecpause 3
/*
/*for the no closure case the tnode becomes the new fnode
/*
&sv fnode = %tnode%
&end
/*
/*-le6030
&end
/*
```

```
/*--- reservoir extension (5.1.3.)---
/*
&if ( %reservoir% = yes ) &then
&do
/*-ls6035
&type HECPREPRO:
&type HECPREPRO: RESERVOIR EXTENSION
&type HECPREPRO:
&r hecpause 4
/*
/*continue downstream
/*
&sv closed2 = .FALSE.
&do &until %closed2%
/*-ls6050
/*
/*go down the AAT and look at all the arcs until suspect is found.
/*since reservoir diversions are supported we have to make sure we
/*don't leave the reservoir on a diversion which would mean we would
/*never find dnreservoir node.
/*
cursor cur6040 declare hydrocov.aat info ro
cursor cur6040 open
&sv eof6040 = .FALSE.
&do &until %eof6040%
/*-ls6060
/*
&sv valid2 = no
&if ( %:cur6040.fnode#% = %fnode% ) &then
&do
&sv valid2 = yes
&if ( ( %:cur6040.rpoly#% = 1 ) and ( %:cur6040.lpoly#% = 1 ) ) &then
   &sv valid2 = no
&end
&if ( %valid2% = yes ) &then
&do
&sv tnode = %:cur6040.tnode#%
&sv arc = %:cur6040.hydrocov#%
&sv eof6040 = .TRUE.
&end
/*
cursor cur6040 next
&if %:cur6040.AML$NEXT% = .FALSE. &then
   &sv eof6040 = .TRUE.
/*-le6060
&end
cursor cur6040 remove
/*
/*
/*if the arcs tnode is a valid element the end of the channel is found
/*and the channel is closed, otherwise the next downstream arc has to be
/*examined.  valid elements are only dnreservoirs (hectype 10).  for each
/*dnreservoir we encounter here we have to add 1 to the h#tnodes variable
/*since it might turn into a junction
/*
&sv nhecid = [ value nhecid%tnode% ]
&sv rnodetype = [ value rnodetype%tnode% ]
&sv hnodetype = [ value hnodetype%tnode% ]
&sv nhectype = [ value nhectype%tnode% ]
&sv h#tnodes = [ value h#tnodes%tnode% ]
&sv valid = no
&sv junction = no
&if ( %nhectype% = 10 ) &then
&do
/*-ls6064
&sv valid = yes
/*
/*update the h#tnodes array
/*
&sv h#tnodes = %h#tnodes% + 1
&sv h#tnodes%tnode% = %h#tnodes%
```

```
&sv hh#tnodes%nhecid% = %h#tnodes%
/*
&if ( %h#tnodes% = 2 ) &then
&do
/*-ls6065
&sv junction = yes
&sv junrun = 2
&sv junrun%tnode% = %junrun%
/*-le6065
&end
&if ( %h#tnodes% > 2 ) &then
&do
/*-ls6066
&sv junction = yes
&sv junrun = [ value junrun%tnode% ]
&sv junrun = %junrun% + 1
&sv junrun%tnode% = %junrun%
/*-le6066
&end
/*-le6064
&end
&type HECPREPRO:
&type HECPREPRO: Searching for reservoir outlet
&type HECPREPRO:
&type HECPREPRO: ELEMENT INFORMATION
&type HECPREPRO:
&type HECPREPRO: TNODE: %tnode%
&type HECPREPRO: HECID: %nhecid%
&type HECPREPRO:
&type HECPREPRO: RNODETYPE: %rnodetype%
&type HECPREPRO: HNODETYPE: %hnodetype%
&type HECPREPRO: H#TNODES: %h#tnodes%
&type HECPREPRO: HECTYPE: %nhectype%
&type HECPREPRO: TNODE VALIDITY: %valid%
&type HECPREPRO:
&if ( %junction% = yes ) &then
&do
&type HECPREPRO: JUNCTION INFORMATION
&type HECPREPRO: JUNRUN: %junrun%
&type HECPREPRO:
&r hecpause 3
&end
/*
/*the reach closes if the node is valid
/*
&if ( %valid% = yes ) &then
&do
&type HECPREPRO:
&type HECPREPRO: RESERVOIR EXTENSION
&type HECPREPRO:
&type HECPREPRO: Extension Closure
&type HECPREPRO:
&r hecpause 3
&sv closed2 = .TRUE.
&end
&if ( %valid% = no ) &then
&do
&type HECPREPRO:
&type HECPREPRO: RESERVOIR EXTENSION
&type HECPREPRO:
&type HECPREPRO: No Extension Closure
&type HECPREPRO:
&r hecpause 3
/*
/*for the no closure case the tnode becomes the new fnode
/*
&sv fnode = %tnode%
&end
/*
/*-le6050
&end
```

```
/*
&sv reservoir = no
/*
/*-le6035
&end
/*
/*--- record reach data (5.1.4.) ---
/*
/*write the HECID to all the arcs of the reach element
/*
calculate hydrocov arc HECID = %hecid%
&type HECPREPRO:
&type HECPREPRO: Recording HECID
&type HECPREPRO:
&type HECPREPRO: ARC(S): %arcs%
&type HECPREPRO: HECID: %hecid%
&type HECPREPRO:
&r hecpause 3
/*
/*write the HECTYPE, HECX, HECY and HECDNID of the reach to array
/*
&sv hectype%hecid% = 4
&sv hecx%hecid% = -1
&sv hecy%hecid% = -1
&sv 1hecdnid%hecid% = [ value nhecid%tnode% ]
/*
/*write the HECUPID to all the arcs of the reach element
/*
&sv hecupid = %:cur6020.hecid%
calculate hydrocov arc HECUPID = %hecupid%
&type HECPREPRO:
&type HECPREPRO: Recording HECUPID
&type HECPREPRO:
&type HECPREPRO: ARC(S): %arcs%
&type HECPREPRO: HECUPID: %hecupid%
&type HECPREPRO:
&r hecpause 3
/*
/*write the HECDNID to all the arcs of the channel element
/*
&sv hecdnid = [ value nhecid%tnode% ]
calculate hydrocov arc HECDNID = %hecdnid%
&type HECPREPRO:
&type HECPREPRO: Recording HECDNID
&type HECPREPRO:
&type HECPREPRO: ARC(S): %arcs%
&type HECPREPRO: HECDNID: %hecdnid%
&type HECPREPRO:
&r hecpause 3
/*
/*write the HECLENGTH and HECSLOPE to all the arcs of the reach element
/*
&if ( %attrib% = yes ) &then
&do
calculate hydrocov arc HECLENGTH = %heclength%
calculate hydrocov arc HECSLOPE = %hecslope%
&type HECPREPRO:
&type HECPREPRO: Recording HECLENGTH and HECSLOPE
&type HECPREPRO:
&type HECPREPRO: ARC(S): %arcs%
&type HECPREPRO: HECLENGTH: %heclength%
&type HECPREPRO: HECSLOPE: %hecslope%
&type HECPREPRO:
&r hecpause 3
&end
/*
/*--- record upstream element data (5.1.5.) ---
/*
/*write the HECDNID to the upstream node
/*
&sv hecdnid = %hecid%
```

```
&s :cur6020.hecdnid = %hecdnid%
/*
/*if we are dealing with a diversion we will record zero as hecdnid
/*
&if ( %diversion% = yes ) &then
&do
&sv hecdnid = 0
&s :cur6020.hecdnid = 0
&end
&type HECPREPRO:
&type HECPREPRO: Recording HECDNID of upstream element
&type HECPREPRO:
&type HECPREPRO: HECID: %:cur6020.hecid%
&type HECPREPRO: HECDNID: %hecdnid%
&type HECPREPRO:
&r hecpause 3
/*
/*write the HECTYPE, HECX, HECY and HECDNID of the upstream node to array
/*
&sv hectype%:cur6020.hecid% = %:cur6020.hectype%
&sv hecx%:cur6020.hecid% = %:cur6020.x-coord%
&sv hecy%:cur6020.hecid% = %:cur6020.y-coord%
&if ( %diversion% = no ) &then
   &sv 1hecdnid%:cur6020.hecid% = %:cur6020.hecdnid%
/*
/*if we are dealing with a diversion the hecdnid will also be written to
/*a matrix
/*
&if ( %diversion% = yes ) &then
&do
   &sv %divrun%hecdnid%:cur6020.hecid% = %hecid%
   &type HECPREPRO:
   &type HECPREPRO: DIVERSION INFORMATION
   &type HECPREPRO: HECID: %:cur6020.hecid%
   &type HECPREPRO: DIVRUN: %divrun%
   &type HECPREPRO: HECDNID: %hecid%
   &type HECPREPRO:
   &r hecpause 3
&end
/*
/*--- record downstream element data (5.1.6.) ---
/*
/*write the HECUPID to the downstream node
/*
&sv dnhecid = [ value nhecid%tnode% ]
&sv hecupid = %hecid%
/*
&if ( %h#tnodes% = 1 ) &then
&do
/*&sv hecupid%dnhecid% = %hecid%
&sv 1hecupid%dnhecid% = %hecid%
&end
/*
/*if we are dealing with a junction we will record zero as hecupid
/*
&if ( %junction% = yes ) &then
&do
&sv hecupid = 0
&sv %junrun%hecupid%dnhecid% = %hecid%
&end
&type HECPREPRO:
&type HECPREPRO: Recording HECUPID of downstream element
&type HECPREPRO:
&type HECPREPRO: HECID: %dnhecid%
&type HECPREPRO: HECUPID: %hecupid%
&type HECPREPRO:
&r hecpause 3
/*
/*write the HECTYPE, HECX, HECY and HECUPID of the downstream node to array
/*
/*&sv hectype%dnhecid% = [ value hectype%dnhecid% ]
```

```
/*&sv hecx%dnhecid% = [ value hecx%dnhecid% ]
/*&sv hecy%dnhecid% = [ value hecy%dnhecid% ]
/*&sv 1hecupid%dnhecid% = %hecid%
/*
/*
/*if we are dealing with a junction the hecupid will also be written to
/*a matrix
/*
&if ( %junction% = yes ) &then
&do
   &sv %junrun%hecupid%dnhecid% = %hecid%
   &type HECPREPRO:
   &type HECPREPRO: JUNCTION INFORMATION
   &type HECPREPRO: HECID: %dnhecid%
   &type HECPREPRO: JUNRUN: %junrun%
   &type HECPREPRO: HECUPID: %hecid%
   &type HECPREPRO:
   &r hecpause 3
&end
/*
/*--- show HECIDs on the screen ---
/*
&if ( %.oblevel% ne 0 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: Labeling channel
&type HECPREPRO:
&r hecpause 3
textcolor 7
arctext hydrocov hecid # LL
&end
/*
/*--- show HECDNIDs on the screen ---
/*
&if ( %.oblevel% ne 0 ) &then
&do
textcolor 5
arctext hydrocov hecdnid # UR
&end
/*
/*--- keep hecid counter going ---
/*
&sv hecid = %hecid% + 1
/*
/*-l6010
&end
/*
/*--- close loop ---
/*
/*note that we have to go back to the same node again if we are at
/*a diversion, which means we will not next on cur6020. then we can't
/*pick the same first arc again.
/*
&if ( ( %diversion% = no ) or ( %divrun% = %h#fnodes% ) ) &then
&do
cursor cur6020 next
&if %:cur6020.AML$NEXT% = .FALSE. &then
   &sv eof6020 = .TRUE.
&end
/*
/*after we are through a diversion we also have to rezero
/*the divrun variable for the next diversion.
/*
&if ( ( %diversion% = yes ) and ( %divrun% = %h#fnodes% ) ) &then
   &sv divrun = 0
/*
/*-le6020
&end
cursor cur6020 remove
/*
/*--- show hecdnid of channel elements on screen ---
```

```
/*
&if ( %.oblevel% ne 0 ) &then
&do
unselect hydrocov nodes
aselect hydrocov nodes ( HECTYPE = 1 )
aselect hydrocov nodes ( HECTYPE = 3 )
aselect hydrocov nodes ( HECTYPE = 6 )
aselect hydrocov nodes ( HECTYPE = 7 )
aselect hydrocov nodes ( HECTYPE = 8 )
aselect hydrocov nodes ( HECTYPE = 9 )
aselect hydrocov nodes ( HECTYPE = 10 )
&type HECPREPRO:
&type HECPREPRO: Labeling channel elements
&type HECPREPRO:
&r hecpause 3
nodetext hydrocov hecdnid # UR
&end
/*
/*----------------------
/*-- Subbasins (5.2.) --
/*----------------------
/*
/*--- mark subbasins ---
/*
&if ( %.oblevel% ne 0 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: Marking subbasins (green box)
&type HECPREPRO:
&r hecpause 3
markersymbol 7
points subcov
&end
/*
/*below only needed if attributes get collected
/*
&if ( %attrib% = yes ) &then
&do
/*-ls6085
/*
/*--- Create Subbasin Point Coverage (5.2.1.) ---
/*
&if [exists subcov2 -cover] &then
&sys arc kill subcov2 all
create subcov2 select subcov point
&sys arc build subcov2 point
&sys arc addxy subcov2 point
&sys arc additem subcov2.pat subcov2.pat INTX 8 12 i
&sys arc additem subcov2.pat subcov2.pat INTY 8 12 i
calculate subcov2 point intx = x-coord
calculate subcov2 point inty = y-coord
&sys arc additem subcov.pat subcov.pat INTX 8 12 i
&sys arc additem subcov.pat subcov.pat INTY 8 12 i
calculate subcov poly intx = x-coord
calculate subcov poly inty = y-coord
/*
/*--- set up hydrocov for network and dynamic segmentation (5.2.2.) ---
/*
&sys arc arcsection hydrocov fpath
&sys arc measureroute arc hydrocov fpath length length
netcover hydrocov fpath
coordinate keyboard xy
/*
/*-le6085
&end
/*
/*--- process each subbasin ---
/*
cursor cur6070 declare subcov.pat info rw
cursor cur6070 open
/*
```

```
/*skip the first universal poly
/*
cursor cur6070 next
/*
&sv eof6070 = .FALSE.
&do &until %eof6070%
/*-ls6090
/*
/*
/*--- record general subbasin data (5.2.3.) ---
/*
/*write HECID and HECTYPE
/*
&s :cur6070.hecid = %hecid%
&s :cur6070.hectype = 5
/*
/*write HECDNID
/*
&sv dnnode = [ value polydnnode%:cur6070.subcov#% ]
&sv hecdnid = [ value nhecid%dnnode% ]
&s :cur6070.hecdnid = %hecdnid%
/*
/*write the HECTYPE, HECX, HECY and HECDNID to array
/*
&sv hectype%:cur6070.hecid% = %:cur6070.hectype%
&sv hecx%:cur6070.hecid% = %:cur6070.x-coord%
&sv hecy%:cur6070.hecid% = %:cur6070.y-coord%
&sv 1hecdnid%:cur6070.hecid% = %:cur6070.hecdnid%
/*
&type HECPREPRO:
&type HECPREPRO: Recording subbasin HECID, HECTYPE and HECDNID
&type HECPREPRO:
&type HECPREPRO: POLY: %:cur6070.subcov#%
&type HECPREPRO: HECID: %hecid%
&type HECPREPRO: HECTYPE: %:cur6070.hectype%
&type HECPREPRO: HECDNID: %hecdnid%
&type HECPREPRO:
&r hecpause 3
/*
/*write the HECUPID to array
/*
/*note that a subbasin outlet can also be a junction
&sv h#tnodes = [ value h#tnodes%dnnode% ]
&sv hh#tnodes%hecdnid% = %h#tnodes%
&sv junction = no
&if ( %h#tnodes% > 1 ) &then
   &sv junction = yes
/*
&if ( %junction% = no ) &then
&do
&sv 1hecupid%:cur6070.hecdnid% = %hecid%
&end
/*
&if ( %junction% = yes ) &then
&do
/*-lsa
&sv i = 1
&sv found = no
&do &while ( %found% = no )
/*-leb
&sv a = %i%hecupid%:cur6070.hecdnid%
&sv b = [ value %a% ]
&if ( [ length %b% ] = 0 ) &then
&do
/*-lec
&sv %i%hecupid%:cur6070.hecdnid% = %hecid%
&sv found = yes
/*-lec
&end
&if ( %i% > %h#tnodes% ) &then
&do
```

```
/*-led
&type HECPREPRO:
&type HECPREPRO: ERROR: SUBBASIN OUTLET - SUBBASIN CONNECTION
&type HECPREPRO:
&r hecpause 1
&sv errors = %errors% + 1
&sv found = yes
/*-led
&end
&sv i = %i% + 1
/*-leb
&end
/*-lea
&end
/*
&type HECPREPRO:
&type HECPREPRO: Recording subbasin outlet HECUPID
&type HECPREPRO:
&type HECPREPRO: HECID: %hecdnid%
&type HECPREPRO: HECUPID: %hecid%
&type HECPREPRO:
&if ( %junction% = yes ) &then
&do
&sv 1hecupid = [ value 1hecupid%:cur6070.hecdnid% ]
&type HECPREPRO:
&type HECPREPRO: JUNCTION INFORMATION
&type HECPREPRO: 1HECUPID: %1hecupid%
&type HECPREPRO: HECUPID: %hecid%
&type HECPREPRO:
&end
&r hecpause 3
/*
&sv hecid = %hecid% + 1
/*
/*attributes are optional
/*
&if ( %attrib% = yes ) &then
&do
/*-ls6095
/*
/*--- write coordinates of subbasin boundary to a file (5.2.4.) ---
/*
&sv b-coord = bc%:cur6070.subcov#%
unselect subcov arcs
aselect subcov arcs ( lpoly# = %:cur6070.subcov#% )
aselect subcov arcs ( rpoly# = %:cur6070.subcov#% )
writeselect file1
aselect subcov arcs
&if [exists bound -cover] &then
&sys arc kill bound all
&sys arc reselect subcov bound arc file1 arc
&sv a = [delete file1 -file]
&if [exists %b-coord%] &then
&sv a = [delete %b-coord% -file]
&sys arc ungenerate line bound %b-coord%
&sys arc kill bound all
&s :cur6070.b-coord = %b-coord%
&type HECPREPRO:
&type HECPREPRO: Recording subbasin B-COORD
&type HECPREPRO:
&type HECPREPRO: POLY: %:cur6070.subcov#%
&type HECPREPRO: B-COORD: %b-coord%
&type HECPREPRO:
&r hecpause 3
/*
/*--- calculate attributes of longest flow path (5.2.5.) ---
/*
&sv longestlength = 0
&sv longestnode = 0
/*select arc belonging to poly
unselect hydrocov arcs
```

```
aselect hydrocov arcs ( subcov# = %:cur6070.subcov#% )
/*select nodes belonging to poly
unselect hydrocov nodes
cursor cur6075 declare hydrocov arc ro
cursor cur6075 open
&sv eof6075 = .FALSE.
&do &until %eof6075%
/*-ls6096
aselect hydrocov node ( hydrocov# = %:cur6075.fnode#% )
aselect hydrocov node ( hydrocov# = %:cur6075.tnode#% )
cursor cur6075 next
&if %:cur6075.AML$NEXT% = .FALSE. &then
   &sv eof6075 = .TRUE.
/*-le6096
&end
cursor cur6075 remove
/*remember nodes belonging to poly
writeselect allnodes
/*compute node to node distance file for all nodes in subbasin
&if [exists distfile -info] &then
&sv a = [delete distfile -info]
nodedistance nodes nodes distfile 1000000
/*get outletnode
&sv outletnode = [ value polydnnode%:cur6070.subcov#% ]
/*select records in distfile originating in outlet node
unselect distfile info
aselect distfile info ( hydrocov#a = %outletnode% )
/*find the longest flow path
cursor cur6077 declare distfile info ro
cursor cur6077 open
&sv eof6077 = .FALSE.
&do &until %eof6077%
/*-ls6099
&if ( %:cur6077.network% > %longestlength% ) &then
&do
&sv longestlength = %:cur6077.network%
&sv longestnode = %:cur6077.hydrocov#b%
&end
cursor cur6077 next
&if %:cur6077.AML$NEXT% = .FALSE. &then
   &sv eof6077 = .TRUE.
/*-le6099
&end
cursor cur6077 remove
/*trace downstream of longestnode and ungenerate coordinates
unselect hydrocov nodes
aselect hydrocov nodes ( hydrocov# = %longestnode% )
&if ( %.oblevel% ne 0 ) &then
&do
markersymbol 19
nodes hydrocov
&end
writeselect upnode
readselect allnodes
&sv a = [delete allnodes -file]
trace downstream hydrocov longestpath # upnode
&sv a = [delete upnode -file]
/*ungenerate longest flow path
&sv f-coord = fc%:cur6070.subcov#%
aselect hydrocov arcs
&if [exists path -cover] &then
&sys arc kill path all
&sys arc reselect hydrocov path arc longestpath arc
&sv a = [delete longestpath -file]
&if [exists %f-coord%] &then
&sv a = [delete %f-coord% -file]
&sys arc ungenerate line path %f-coord%
&s :cur6070.f-coord = %f-coord%
/*record elevation of longest flow path
&sv fupz = [ value nodez%longestnode% ]
&s :cur6070.fupz = %fupz%
```

```
/*record length of longest flow path
&s :cur6070.flength = %longestlength%
/*
&type HECPREPRO:
&type HECPREPRO: Recording F-COORD, FUPZ and FLENGTH
&type HECPREPRO:
&type HECPREPRO: POLY: %:cur6070.subcov#%
&type HECPREPRO:
&type HECPREPRO: F-COORD: %f-coord%
&type HECPREPRO: FUPZ: %fupz%
&type HECPREPRO: FLENGTH: %longestlength%
&type HECPREPRO:
&r hecpause 3
/*
/*--- calculate length along longest flow path from point --------
/*--- nearest to subbasin centroid to subbasin outlet (5.2.6.) ---
/*
/*do a near from subcov2 to path
&sys arc near subcov2 path line 1000000 # location
/*get xy of nearest point on stream and nearest stream arc
unselect subcov2 points
aselect subcov2 points ( intx = %:cur6070.intx% )
unselect subcov2 points ( inty ne %:cur6070.inty% )
&sv nearx = [show select subcov2 point 1 item x-coord]
&sv neary = [show select subcov2 point 1 item y-coord]
/*get hydrocov# of nearest downstream node
unselect hydrocov arcs
aselect hydrocov arcs one *
%nearx%, %neary%
&sv tnode = [show select hydrocov line 1 item tnode#]
/*get flow length from arcs tnode to outlet
unselect distfile info ( hydrocov#b ne %tnode% )
&sv flength2a = [show select distfile info 1 item network]
/*get flow length from nearest point to arcs tnode
measure route hydrocov fpath
%nearx%, %neary%
&sv flength2b = [extract 5 [show measure route]]
/*calculate flength2
&sv flength2 = %flength2a% + %flength2b%
/*record flength2
&s :cur6070.flength2 = %flength2%
/*
&type HECPREPRO:
&type HECPREPRO: Recording FLENGTH2
&type HECPREPRO:
&type HECPREPRO: POLY: %:cur6070.subcov#%
&type HECPREPRO:
&type HECPREPRO: FLENGTH2a: %flength2a%
&type HECPREPRO: FLENGTH2b: %flength2b%
&type HECPREPRO: FLENGTH2: %flength2%
&type HECPREPRO:
&r hecpause 3
/*
&sys arc kill path all
/*
/*-le6095
&end
/*
cursor cur6070 next
&if %:cur6070.AML$NEXT% = .FALSE. &then
   &sv eof6070 = .TRUE.
/*-le6090
&end
cursor cur6070 remove
/*
/*label subbasins
/*
&if ( %.oblevel% ne 0 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: Labeling subbasins
```

```
&type HECPREPRO:
&r hecpause 3
textcolor 3
pointtext subcov hecid # LL
textcolor 5
pointtext subcov hecdnid # UR
&end
/*
/*---------------------------------
/*-- Reservoir Diversions (5.3.) --
/*---------------------------------
/*
/*only needs to be done if there are reservoirs
/*
&if ( %reservoirs% = yes ) &then
&do
/*-ls60100b
/*
/*only needs to be done on selective set of nodes
/*
unselect hydrocov.nat info ( hectype ne 4 )
/*
/*--- find reservoir diversion (5.3.1.) ---
/*
/*process each node
/*
cursor cur6080 declare hydrocov.nat info rw
cursor cur6080 open
&sv eof6080 = .FALSE.
&do &until %eof6080%
/*-ls60100
/*
/*check node for diversion reservoir
/*
&sv hnodetype = [ value hnodetype%:cur6080.hydrocov#% ]
&if ( %hnodetype% = dreservoir ) &then
&do
/*-ls60110
/*
&type HECPREPRO:
&type HECPREPRO: Searching for reservoir diversion
&type HECPREPRO:
&type HECPREPRO: DIVERSION NODE INFORMATION
&type HECPREPRO: NODE: %:cur6080.hydrocov#%
&type HECPREPRO:
&r hecpause 3
/*
/*--- find downstream reach (5.3.2.) ---
/*
aselect hydrocov.aat info
cursor cur6090 declare hydrocov.aat info ro
cursor cur6090 open
&sv eof6090 = .FALSE.
&do &until %eof6090%
/*-ls60120
/*
/*for the downstream non reservoir arc only
/*
&sv valid = no
&if ( %:cur6090.fnode#% = %:cur6080.hydrocov#% ) &then
&do
&if ( ( %:cur6090.rpoly#% = 1 ) and ( %:cur6090.lpoly#% = 1 ) ) &then
&sv valid = yes
&end
/*
&if ( %valid% = yes ) &then
&do
&sv rhecid = %:cur6090.hecid%
&type HECPREPRO:
&type HECPREPRO: Searching for reservoir diversion reach
&type HECPREPRO:
```

```
&type HECPREPRO: DIVERSION NODE INFORMATION
&type HECPREPRO: NODE: %:cur6080.hydrocov#%
&type HECPREPRO:
&type HECPREPRO: DIVERSION REACH INFORMATION
&type HECPREPRO: ARC: %:cur6090.hydrocov#%
&type HECPREPRO: HECID: %rhecid%
&type HECPREPRO: FNODE: %:cur6090.fnode#%
&type HECPREPRO: RPOLY: %:cur6090.rpoly#%
&type HECPREPRO: LPOLY: %:cur6090.lpoly#%
&type HECPREPRO:
&r hecpause 3
/*
&end
/*
cursor cur6090 next
&if %:cur6090.AML$NEXT% = .FALSE. &then
   &sv eof6090 = .TRUE.
/*-le60120
&end
cursor cur6090 remove
/*
/*--- find reservoir element (5.3.3.) ---
/*
cursor cur60100 declare hydrocov.aat info rw
cursor cur60100 open
&sv eof60100 = .FALSE.
&do &until %eof60100%
/*-ls60130
/*
/*for the upstream arc only
/*
&if ( %:cur60100.tnode#% = %:cur6080.hydrocov#% ) &then
&do
/*-ls60140
/*
/*get the poly of the upstream arc
/*
&if ( %:cur60100.rpoly#% ne 1 ) &then
&sv poly = %:cur60100.rpoly#%
&if ( %:cur60100.lpoly#% ne 1 ) &then
&sv poly = %:cur60100.lpoly#%
/*
&type HECPREPRO:
&type HECPREPRO: Searching for diversion reservoir
&type HECPREPRO:
&type HECPREPRO: DIVERSION NODE INFORMATION
&type HECPREPRO: NODE: %:cur6080.hydrocov#%
&type HECPREPRO:
&type HECPREPRO: DIVERSION REACH INFORMATION
&type HECPREPRO: HECID: %rhecid%
&type HECPREPRO:
&type HECPREPRO: RESERVOIR INFORMATION
&type HECPREPRO: POLY: %poly%
&type HECPREPRO:
&r hecpause 3
/*
/*select and mark arcs with this poly
/*
unselect hydrocov node
unselect hydrocov arc
aselect hydrocov arc ( rpoly# = %poly% )
aselect hydrocov arc ( lpoly# = %poly% )
&if ( %.oblevel% ne 0 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: Marking diversion reservoir (green)
&type HECPREPRO:
&r hecpause 3
linecolor 3
arcs hydrocov
&end
```

```
/*
/*cursor on those arcs
/*
cursor cur60110 declare hydrocov arc ro
cursor cur60110 open
&sv eof60110 = .FALSE.
&do &until %eof60110%
/*-ls60150
/*
/*find the dnreservoir
/*
&sv thnodetype = [ value hnodetype%:cur60110.tnode#% ]
&sv fhnodetype = [ value hnodetype%:cur60110.fnode#% ]
&if ( ( %thnodetype% = dnreservoir ) or ( %fhnodetype% = dnreservoir ) ) &then
&do
/*-ls60160
&if ( %thnodetype% = dnreservoir ) &then
   &do
   &sv dnhnodetype = %thnodetype%
   &sv dnnode = %:cur60110.tnode#%
   &sv dnhecid = [ value nhecid%:cur60110.tnode#% ]
   &sv dnhecdnid = [ value 1hecdnid%dnhecid% ]
   &sv dnhectype = [ value hectype%dnhecid% ]
   &sv dnh#fnodes = [ value h#fnodes%:cur60110.tnode#% ]
   &end
&if ( %fhnodetype% = dnreservoir ) &then
   &do
   &sv dnhnodetype = %fhnodetype%
   &sv dnnode = %:cur60110.fnode#%
   &sv dnhecid = [ value nhecid%:cur60110.fnode#% ]
   &sv dnhecdnid = [ value 1hecdnid%dnhecid% ]
   &sv dnhectype = [ value hectype%dnhecid% ]
   &sv dnh#fnodes = [ value h#fnodes%:cur60110.fnode#% ]
   &end
/*
/*as of now the dnreservoir is a diversion.  if this was not a diversion
/*previously (h#fnodes = 1) we have to write the old hecdnid%hecid%
/*to 1hecdnid%hecid%.
/*
&if ( %dnh#fnodes% = 1 ) &then
   &sv 1hecdnid%dnhecid% = [ value 1hecdnid%dnhecid% ]
/*
/*update the h#fnodes and hh#fnodes variables
/*
&sv newh#fnodes = %dnh#fnodes% + 1
&sv hh#fnodes%dnhecid% = %newh#fnodes%
&sv h#fnodes%dnnode% = %newh#fnodes%
/*
/*the hecid of the dreservoir has to be written to the
/*%div%hecdnid%hecid% matrix
/*
&sv %newh#fnodes%hecdnid%dnhecid% = %rhecid%
/*
/*--- record data (5.3.4.) ---
/*
/*the hecupid of the reach has to be written to the
/*AAT and 1hecupid%hecid% array
/*
&s :cur60100.hecupid = %dnhecid%
&sv 1hecupid%rhecid% = %dnhecid%
/*
/*
&type HECPREPRO:
&type HECPREPRO: Searching for reservoir outlet
&type HECPREPRO:
&type HECPREPRO: DIVERSION NODE INFORMATION
&type HECPREPRO: NODE: %:cur6080.hydrocov#%
&type HECPREPRO:
&type HECPREPRO: DIVERSION REACH INFORMATION
&type HECPREPRO: HECID: %rhecid%
&type HECPREPRO: HECUPID: %dnhecid%
```

```
&type HECPREPRO:
&type HECPREPRO: RESERVOIR INFORMATION
&type HECPREPRO: POLY: %poly%
&type HECPREPRO:
&type HECPREPRO: RESERVOIR OUTLET INFORMATION
&type HECPREPRO: NODE: %dnnode%
&type HECPREPRO: HECID: %dnhecid%
&type HECPREPRO: HECTYPE: %dnhectype%
&type HECPREPRO: H#FNODES: %newh#fnodes%
&type HECPREPRO: 1HECDNID: %dnhecdnid%
&type HECPREPRO: (LAST)HECDNID: %rhecid%
&type HECPREPRO:
&r hecpause 3
/*
/*kick out of loop
/*
&sv eof60110 = .TRUE.
/*-le60160
&end
/*
cursor cur60110 next
&if %:cur60110.AML$NEXT% = .FALSE. &then
   &sv eof60110 = .TRUE.
/*
/*-le60150
&end
cursor cur60110 remove
/*
/*-le60140
&end
/*
cursor cur60100 next
&if %:cur60100.AML$NEXT% = .FALSE. &then
   &sv eof60100 = .TRUE.
/*-le60130
&end
cursor cur60100 remove
/*
/*restore cyan color of reservoir
/*
&if ( %.oblevel% ne 0 ) &then
&do
linecolor 5
arcs hydrocov
&end
/*
/*-le60110
&end
/*
cursor cur6080 next
&if %:cur6080.AML$NEXT% = .FALSE. &then
   &sv eof6080 = .TRUE.
/*-le60100
&end
cursor cur6080 remove
/*
/*-le60100b
&end
/*
/*----------------------------------
/*--- Allow for User Examination ---
/*----------------------------------
/*
&type
&type HECPREPRO:
&type HECPREPRO: HYDROCOV LEGEND:
&type HECPREPRO: ------------------------------------------------
&type HECPREPRO:
&type HECPREPRO: LINES:
&type HECPREPRO: White:  Subbasin Boundary and Streams Outside
&type HECPREPRO:         Watershed Boundary
```

```
&type HECPREPRO: Yellow: Channel System
&type HECPREPRO: Red:    Non Channel System
&type HECPREPRO: Cyan:   Reservoir Shoreline
&type HECPREPRO:
&type HECPREPRO: SYMBOLS:
&type HECPREPRO: Green Triangles:           Subbasin Outlets
&type HECPREPRO: Red Triangles:             Sinks
&type HECPREPRO: Blue Triangles:            Sources
&type HECPREPRO: Green Circles:             Junctions
&type HECPREPRO: Blue Circles:              Diversions
&type HECPREPRO: Green Boxes:               Subbasins
&type HECPREPRO: Upsidedown Green Triangle: Reservoirs
&if ( %attrib% = yes ) &then
&do
&type HECPREPRO: Blue Dot:   Upstream End of Longest Flow Path
&type HECPREPRO: Yellow Box: Point on Longest Flow Path closest to
&type HECPREPRO:             Subbasin Centroid.
&end
&type HECPREPRO:
&type HECPREPRO: TEXT:
&type HECPREPRO: White:  HECID of Channel Elements
&type HECPREPRO: Yellow: HECID of Reaches
&type HECPREPRO: Cyan:   HECDNID of Elements
&type HECPREPRO:
&type HECPREPRO: ------------------------------------------------
&type HECPREPRO:
&type
&r hecpause 2
/*
/*leave arcplot
&if ( %.oblevel% ne 0 ) &then
display 0
q
/*
/*----------------------------------------
/*----------------------------------------
/*--- Step 6. Create Symbolic Coverage ----
/*----------------------------------------
/*----------------------------------------
/*
&type HECPREPRO:
&type HECPREPRO: Step 6. Create Symbolic Coverage
&type HECPREPRO:
&if ( %.oblevel% = .70 ) &then
&r hecpause 0
/*
/*
/*create symcov
&if [exists symcov -cover] &then
kill symcov all
create symcov hydrocov
/*
/*create symcov topology
build symcov node
build symcov line
/*
/*add hecid item
additem symcov.nat symcov.nat hecid 4 4 i #
additem symcov.aat symcov.aat hecid 4 4 i #
additem symcov.aat symcov.aat uphecid 4 4 i #
additem symcov.aat symcov.aat dnhecid 4 4 i #
/*
/*enter arcedit
arcedit
&if [extract 1 [show display]] ne 0 &then
display 0
coordinate keyboard xy
edit symcov
drawenv arc on
editfeature arc
nodesnap closest 1
```

```
setincrement 1 0
autoincrement on
/*
/*-----------------------------
/*--- Channel System (6.1.) ---
/*-----------------------------
/*
/*start with hecid = 1
&sv elements = %hecid% - 1
&type HECPREPRO:
&type HECPREPRO: CHANNEL SYSTEM
&type HECPREPRO: ELEMENTS: %elements%
&type HECPREPRO:
&r hecpause 3
&sv divrun = 0
&sv hecid = 1
&do &until ( %hecid% > %elements% )
/*-ls7010
/*
/*--- find upstream element (6.1.1.) ---
/*
/*pull values
&sv hectype = [ value hectype%hecid% ]
&sv hecx = [ value hecx%hecid% ]
&sv hecy = [ value hecy%hecid% ]
&sv hecdnid = [ value 1hecdnid%hecid% ]
&sv h#fnodes = [ value hh#fnodes%hecid% ]
/*
&type HECPREPRO:
&type HECPREPRO: CURRENT ELEMENT:
&type HECPREPRO: HECID: %hecid% HECTYPE: %hectype% HECDNID: %hecdnid%
&type HECPREPRO: X: %hecx% Y: %hecy%
&type HECPREPRO:
&r hecpause 4
/*
/*an arc will be added if the hectype is a valid upstream element
/*and we have to remember to come back to the same element if it
/*is a diversion (hectype 8) or a reservoir (hectype 10) with more that
/*one h#fnodes.
&sv valid = no
&sv diversion = no
&if ( %hectype% = 1 ) &then
   &sv valid = yes
&if ( %hectype% = 3 ) &then
   &sv valid = yes
&if ( %hectype% = 7 ) &then
   &sv valid = yes
&if ( %hectype% = 8 ) &then
&do
   &sv valid = yes
   &sv diversion = yes
   &sv divrun = %divrun% + 1
&end
&if ( %hectype% = 10 ) &then
&do
/*-ls7015
&sv valid = yes
&if ( %h#fnodes% >= 2 ) &then
&do
   &sv diversion = yes
   &sv divrun = %divrun% + 1
&end
/*-le7015
&end
/*
&type HECPREPRO:
&type HECPREPRO: UPSTREAM NODE VALIDITY: %valid%
&type HECPREPRO:
&r hecpause 4
/*
&if ( %valid% = yes ) &then
```

```
&do
/*-ls7020
/*if we are dealing with a diversion an adjustment has to be made
/*to hecdnid, because the value stored in the hecdnid%hecid% is zero
&if ( %diversion% = yes ) &then
&do
/*-ls7030
/*
/*retrieving data from a matrix has to be done in a two step as below
&sv matrix = %divrun%hecdnid%hecid%
&sv hecdnid = [ value %matrix% ]
/*
&type HECPREPRO: DIVERSION INFORMATION
&type HECPREPRO: DIVRUN: %divrun% HECDNID: %hecdnid%
&type HECPREPRO:
&r hecpause 4
/*-le7030
&end
/*
/*--- find reach (6.1.2.) ---
/*
/*pull values for the first downstream element
&sv dn1hectype = [ value hectype%hecdnid% ]
&sv dn1hecx = [ value hecx%hecdnid% ]
&sv dn1hecy = [ value hecy%hecdnid% ]
&sv dn1hecdnid = [ value 1hecdnid%hecdnid% ]
/*
&type HECPREPRO:
&type HECPREPRO: FIRST DOWNSTREAM ELEMENT:
&type HECPREPRO: HECID: %hecdnid% HECTYPE: %dn1hectype% HECDNID: %dn1hecdnid%
&type HECPREPRO:
&r hecpause 4
/*
/*an error has occured if the first downstream element is not a hectype 4
&if ( %dn1hectype% ne 4 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: NO CHANNEL VALIDITY - ERROR CONDITION
&type HECPREPRO:
&r hecpause 1
&sv errors = %errors% + 1
&end
/*
&if ( %dn1hectype% = 4 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: CHANNEL VALIDITY
&type HECPREPRO:
&r hecpause 4
&end
/*
/*--- find downstream element (6.1.3.) ---
/*
/*pull values for the second downstream element
&sv dn2hectype = [ value hectype%dn1hecdnid% ]
&sv dn2hecx = [ value hecx%dn1hecdnid% ]
&sv dn2hecy = [ value hecy%dn1hecdnid% ]
&sv dn2hecdnid = [ value 1hecdnid%dn1hecdnid% ]
/*
&type HECPREPRO:
&type HECPREPRO: SECOND DOWNSTREAM ELEMENT:
&type HECPREPRO: HECID: %dn1hecdnid% HECTYPE: %dn2hectype% HECDNID:
%dn2hecdnid%
&type HECPREPRO: X: %dn2hecx% Y: %dn2hecy%
&type HECPREPRO:
&r hecpause 4
/*
/*check if the second downstream element is valid
&sv valid = no
&if ( %dn2hectype% = 1 ) &then
&sv valid = yes
```

```
&if ( %dn2hectype% = 3 ) &then
&sv valid = yes
&if ( %dn2hectype% = 6 ) &then
&sv valid = yes
&if ( %dn2hectype% = 8 ) &then
&sv valid = yes
&if ( %dn2hectype% = 10 ) &then
&sv valid = yes
/*
&if ( %valid% = no ) &then
&do
&type HECPREPRO:
&type HECPREPRO: NO DOWNSTREAM NODE VALIDITY - ERROR CONDITION
&type HECPREPRO:
&r hecpause 1
&sv errors = %errors% + 1
&end
/*
&if ( %valid% = yes )  &then
&do
&type HECPREPRO:
&type HECPREPRO: DOWNSTREAM NODE VALIDITY
&type HECPREPRO:
&r hecpause 4
&end
/*
/*--- add line (6.1.4.) ---
/*
/*summarize data
&type HECPREPRO:
&type HECPREPRO: ADDING ELEMENT: HECID: %hecdnid%
&type HECPREPRO:   FROM ELEMENT: HECID: %hecid% X: %hecx% Y: %hecy%
&type HECPREPRO:     TO ELEMENT: HECID: %dn1hecdnid% X: %dn2hecx% Y: %dn2hecy%
&type HECPREPRO:
&r hecpause 3
/*add arc
add
2, %hecx%, %hecy%
2, %dn2hecx%, %dn2hecy%
9
/*add hecid, hectype, uphecid and dnhecid to AAT
calculate hecid = %hecdnid%
calculate uphecid = %hecid%
calculate dnhecid = %dn1hecdnid%
/*-le7020
&end
/*
/*we have to go back to the same element once more if we are dealing
/*with a diversion.  keep in mind that reaches and subbasins are now
/*in the loop and they don't have h#fnodes defined.
&if ( ( %hectype% = 4 ) or ( %hectype% = 5 ) ) &then
   &sv hecid = %hecid% + 1
/*
&if ( ( %hectype% ne 4 ) and ( %hectype% ne 5 ) ) &then
&do
&if ( ( %diversion% = no ) or ( %divrun% = %h#fnodes% ) ) &then
   &sv hecid = %hecid% + 1
&end
/*
/*after the second run through a diversion we have to rezero
/*the run variable.
&if ( ( %hectype% ne 4 ) and ( %hectype% ne 5 ) ) &then
&do
&if ( ( %diversion% = yes ) and ( %divrun% = %h#fnodes% ) ) &then
   &sv divrun = 0
&end
/*
/*-le7010
&end
/*
/*-------------------------
```

```
/*--- Subbasins (6.2.) ---
/*------------------------
/*
/*start with hecid = 1
&type HECPREPRO:
&type HECPREPRO: SUBBASINS
&type HECPREPRO: ELEMENTS: %elements%
&type HECPREPRO:
&r hecpause 3
&sv hecid = 1
&do &until ( %hecid% > %elements% )
/*-ls7040
/*
/*--- find subbasin element (6.2.1.) ---
/*
/*pull values
&sv hectype = [ value hectype%hecid% ]
&sv hecx = [ value hecx%hecid% ]
&sv hecy = [ value hecy%hecid% ]
&sv hecdnid = [ value 1hecdnid%hecid% ]
/*
&type HECPREPRO:
&type HECPREPRO: CURRENT ELEMENT:
&type HECPREPRO: HECID: %hecid% HECTYPE: %hectype% HECDNID: %hecdnid%
&type HECPREPRO: X: %hecx% Y: %hecy%
&type HECPREPRO:
&r hecpause 3
/*
/*an arc will be added if the hectype is a valid upstream element
&sv valid = no
&if ( %hectype% = 5 ) &then
&sv valid = yes
&if ( %valid% = yes ) &then
&do
/*-ls7050
&type HECPREPRO:
&type HECPREPRO: UPSTREAM NODE VALIDITY
&type HECPREPRO:
&r hecpause 3
/*
/*--- find subbasin outlet element (6.2.2.) ---
/*
/*pull values for the downstream element
&sv dn1hectype = [ value hectype%hecdnid% ]
&sv dn1hecx = [ value hecx%hecdnid% ]
&sv dn1hecy = [ value hecy%hecdnid% ]
&sv dn1hecdnid = [ value 1hecdnid%hecdnid% ]
/*
&type HECPREPRO:
&type HECPREPRO: DOWNSTREAM ELEMENT:
&type HECPREPRO: HECID: %hecdnid% HECTYPE: %dn1hectype% HECDNID: %dn1hecdnid%
&type HECPREPRO: X: %dn1hecx% Y: %dn1hecy%
&type HECPREPRO:
&r hecpause 3
/*
/*an error has occured if the downstream element is not a hectype 1
&if ( ( %dn1hectype% ne 1 ) and ( %dn1hectype% ne 6 ) ) &then
&do
&type HECPREPRO:
&type HECPREPRO: NO OUTLET VALIDITY - ERROR CONDITION
&type HECPREPRO:
&r hecpause 1
&end
/*
&if  ( ( %dn1hectype% = 1 ) or ( %dn1hectype% = 1 ) ) &then
&do
&type HECPREPRO:
&type HECPREPRO: OUTLET VALIDITY
&type HECPREPRO:
&r hecpause 3
&end
```

```
/*
/*--- add line (6.2.3.) ---
/*
/*summarize data
&type HECPREPRO:
&type HECPREPRO: ADDING ELEMENT: N/A
&type HECPREPRO:   FROM ELEMENT: HECID: %hecid% X: %hecx% Y: %hecy%
&type HECPREPRO:     TO ELEMENT: HECID: %hecdnid% X: %dn1hecx% Y: %dn1hecy%
&type HECPREPRO:
&r hecpause 3
/*add arc
add
2, %hecx%, %hecy%
2, %dn1hecx%, %dn1hecy%
9
/*add hecid, uphecid and dnhecid to AAT
calculate hecid = 0
calculate uphecid = %hecid%
calculate dnhecid = %hecdnid%
/*-le7050
&end
&sv hecid = %hecid% + 1
/*-le7040
&end
/*
/*
/*quit arcedit
end
y
y
/*
/*--------------------------------
/*--- Update Symcov.nat (6.3.) ---
/*--------------------------------
/*
/*--- update topology ---
/*
build symcov node
build symcov arc
/*
/*--- create hecid2%node% and hectype2%node% arrays ---
/*
cursor cur7010 declare symcov.aat info ro
cursor cur7010 open
/*
&sv eof7010 = .FALSE.
&do &until %eof7010%
/*-ls7060
/*
&sv hecid2%:cur7010.fnode#% = %:cur7010.uphecid%
&type HECPREPRO:
&type HECPREPRO: Updating HECID array
&type HECPREPRO:
&type HECPREPRO: NODE: %:cur7010.fnode#%
&type HECPREPRO: HECID: %:cur7010.uphecid%
&type HECPREPRO:
&r hecpause 4
/*
&sv hecid2%:cur7010.tnode#% = %:cur7010.dnhecid%
&type HECPREPRO:
&type HECPREPRO: Updating HECID array
&type HECPREPRO:
&type HECPREPRO: NODE: %:cur7010.tnode#%
&type HECPREPRO: HECID: %:cur7010.dnhecid%
&type HECPREPRO:
&r hecpause 4
/*
cursor cur7010 next
&if %:cur7010.AML$NEXT% = .FALSE. &then
   &sv eof7010 = .TRUE.
/*-le7060
```

```
&end
cursor cur7010 remove
/*
/*--- update NAT ---
/*
cursor cur7020 declare symcov.nat info rw
cursor cur7020 open
&sv eof7020 = .FALSE.
&do &until %eof7020%
/*-le7070
/*
&sv hecid2 = [ value hecid2%:cur7020.symcov#% ]
&s :cur7020.hecid = %hecid2%
&type HECPREPRO:
&type HECPREPRO: Recording HECID
&type HECPREPRO:
&type HECPREPRO: NODE: %:cur7020.symcov#%
&type HECPREPRO: HECID: %hecid2%
&type HECPREPRO:
&r hecpause 4
/*
cursor cur7020 next
&if %:cur7020.AML$NEXT% = .FALSE. &then
   &sv eof7020 = .TRUE.
/*-le7070
&end
cursor cur7020 remove
/*
/*-----------------------------------------------------
/*-----------------------------------------------------
/*--- Step 7. Relate Attributes to Symbolic Coverage -----
/*-----------------------------------------------------
/*-----------------------------------------------------
/*
&type HECPREPRO:
&type HECPREPRO: Step 7. Relate Attributes to Symbolic Coverage
&type HECPREPRO:
&if ( %.oblevel% = .80 ) &then
&r hecpause 0
/*
/*----------------------------------
/*--- Symcov.aat to Hydrocov.aat ---
/*----------------------------------
/*
relate add
hydroaat
hydrocov.aat
info
hecid
hecid
linear
ro
~
/*
/*----------------------------------
/*--- Symcov.nat to Hydrocov.nat ---
/*----------------------------------
/*
relate add
hydronat
hydrocov.nat
info
hecid
hecid
linear
ro
~
/*
/*--------------------------------
/*--- Symcov.nat to Subcov.pat ---
/*--------------------------------
```

```
/*
relate add
subpat
subcov.pat
info
hecid
hecid
linear
ro
~
/*
/*
/*------------------------------
/*------------------------------
/*--- Step 8. Create Output -----
/*------------------------------
/*------------------------------
/*
&type HECPREPRO:
&type HECPREPRO: Step 8. Create Output
&type HECPREPRO:
&if ( %.oblevel% = .90 ) &then
&r hecpause 0
/*
/*------------------------------
/*--- Create DXF File (8.1) ---
/*------------------------------
/*
&if ( %dxffile% = yes ) &then
&do
/*-ls9002
&type HECPREPRO:
&type HECPREPRO: Creating DXF file
&type HECPREPRO:
&r hecpause 4
/*
/*--- create combined coverage (8.1.1.) ---
/*
/*make temporary coverages
/*
&if [exists streamcov2 -cover] &then
kill streamcov2 all
copy streamcov streamcov2
/*
&if [exists subcov2 -cover] &then
kill subcov2 all
copy subcov subcov2
/*
/*delete AATs
/*
&data arc tables
/*-ls9005
select streamcov2.aat
erase streamcov2.aat
y
select subcov2.aat
erase subcov2.aat
y
end
/*-le9005
&end
/*
/*create AATs
/*
build streamcov2 line
/*
build subcov2 line
/*
/*add layer item to AATs
/*
additem streamcov2.aat streamcov2.aat dxf-layer 16 16 c
```

```
/*
additem subcov2.aat subcov2.aat dxf-layer 16 16 c
/*
/*assign values to layer items
/*
&data arc arcplot
/*-ls9007
calculate streamcov2.aat info dxf-layer = 'streams'
calculate subcov2.aat info dxf-layer = 'subbasins'
q
/*-le9007
&end
/*
/*create combined coverage
/*
&if [exists dxfcov -cover] &then
kill dxfcov all
/*
append dxfcov line none
streamcov2
subcov2
~
y
y
/*
/*kill temporary coverages
/*
kill streamcov2 all
/*
kill subcov2 all
/*
/*--- create dxf file (8.1.2.) ---
/*
&if [exists %dxfname%] &then
&sv a = [delete %dxfname% -file]
arcdxf %dxfname% dxfcov # #
/*
/*kill combined coverage
/*
kill dxfcov all
/*
/*-le90020
&end
/*
/*-------------------------------------------
/*--- Enter Arcplot and Put Data to Screen ---
/*-------------------------------------------
/*
arcplot
&if ( %.oblevel% ne 0 ) &then
&do
&if [extract 1 [show display]] ne 9999 &then
display 9999
&end
/*
/*if user observation level is 4 we allow for change
/*in screen size to get a better look
&if ( %.oblevel% = 4 ) &then
&do
&type HECPREPRO:
&type HECPREPRO: Oportunity to change the display window size
&type HECPREPRO:
&r hecpause 4
&end
/*
mape subcov
&if ( %.oblevel% ne 0 ) &then
&do
linecolor 1
arcs streamcov
arcs subcov
```

```
linecolor 7
arcs symcov
&end
/*
/*-------------------------------------------------------
/*--- Create General Output and HMS Basin File (8.2.) ---
/*-------------------------------------------------------
/*
&type HECPREPRO:
&type HECPREPRO: Creating general output and HMS basin file
&type HECPREPRO:
&r hecpause 4
/*
/*--- set up general output file ---
/*
&sv a = [ close 1 ]
&sv output = [ open %outfile% status -write ]
&sv a = 'HECPREPRO: OUTPUT FILE'
&sv b = [write %output% %a%]
&sv a = 'Created '[date -ufull]
&sv b = [write %output% %a%]
/*
/*--- set up HMS basin file ---
/*
&sv a = [ close 2 ]
&sv hmsbasin = [ open %hmsname% status -write ]
&sv a = Basin: HECPREPRO generated HMS basin file
&sv b = [write %hmsbasin% [quote %a%]]
&sv a = '     Description: HECPREPRO generated HMS basin file'
&sv b = [write %hmsbasin% %a%]
&sv a = '     Last Modified Date: '[date -day]' '[date -month]' '[date -year]
&sv b = [write %hmsbasin% %a%]
&sv hr = [before [date -time] .]
&sv min = [before [after [date -time] .] .]
&sv a = '     Last Modified Time: '%hr%':'%min%
&sv b = [write %hmsbasin% %a%]
&sv a = '     Unit System: Unknown'
&sv b = [write %hmsbasin% %a%]
&sv a = 'End:'
&sv b = [write %hmsbasin% %a%]
&sv a = ''
&sv b = [write %hmsbasin% %a%]
/*
/*--- subbasins ---
/*
/*heading
/*
&sv b = [write %output% '-----------------']
&sv b = [write %output% '--- SUBBASINS ---']
&sv b = [write %output% '-----------------']
&type HECPREPRO:
&type HECPREPRO: SUBBASINS:
&type HECPREPRO:
/*
/*select subbasins
/*
unselect symcov nodes
aselect symcov nodes ( subpat//hectype = 5 )
&if ( %.oblevel% ne 0 ) &then
&do
markersymbol 7
nodes symcov
textcolor 7
nodetext symcov hecid
&end
/*
/*process each subbasin
/*
cursor cur9010 declare symcov node ro
cursor cur9010 open
&sv eof9010 = .FALSE.
```

```
&if %:cur9010.AML$NEXT% = .TRUE. &then
&do
/*-ls9010
&do &until %eof9010%
/*-ls9020
/*
/*write to general text file
/*
&sv b = [write %output% ---]
/*
&type HECPREPRO:
&sv a = HECID:
&sv b = [write %output% %a%]
&sv c = %:cur9010.hecid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*attributes
&if ( %attrib% = 'yes' ) &then
&do
/*-ls9025
/*
&type HECPREPRO:
&sv a = X-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//x-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = Y-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//y-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = CENZ:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//cenz%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = B-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//b-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = F-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//f-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = FUPZ:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//fupz%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
```

```
&type HECPREPRO:
&sv a = FLENGTH:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//flength%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = FLENGTH2:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//flength2%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = AREA:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//area%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = MEANS:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//means%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = MEDIANS:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//medians%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*-le9025
&end
/*
&type HECPREPRO:
&sv a = HECDNID:
&sv b = [write %output% %a%]
&sv c = %:cur9010.subpat//hecdnid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*write to HMS basin file
/*
&if ( %hmsfile% = 'yes' ) &then
&do
/*ls-
/*
&type HECPREPRO:
&type HECPREPRO: Writing to HMS basin file
&type HECPREPRO:
/*
&sv a = 'Subbasin: '
&sv b = %:cur9010.hecid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas X: '
&sv b = %:cur9010.subpat//x-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas Y: '
&sv b = %:cur9010.subpat//y-coord%
```

```
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Downstream: '
&sv b = %:cur9010.subpat//hecdnid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Area: '
&sv b = %:cur9010.subpat//area%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = 'End:'
&sv b = [write %hmsbasin% %a%]
&sv a = ''
&sv b = [write %hmsbasin% %a%]
/*
/*-le
&end
/*
&r hecpause 3
cursor cur9010 next
&if %:cur9010.AML$NEXT% = .FALSE. &then
   &sv eof9010 = .TRUE.
/*-le9020
&end
/*-le9010
&end
cursor cur9010 remove
/*
/*--- sources ---
/*
/*heading
/*
&sv b = [ write %output% '---------------' ]
&sv b = [ write %output% '--- SOURCES ---' ]
&sv b = [ write %output% '---------------' ]
&type HECPREPRO:
&type HECPREPRO: SOURCES:
&type HECPREPRO:
/*
/*select sources
/*
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 7 )
&if ( %.oblevel% ne 0 ) &then
&do
markersymbol 47
nodes symcov
textcolor 7
nodetext symcov hecid
&end
/*
/*write to general text file
/*
cursor cur9020 declare symcov node ro
cursor cur9020 open
&sv eof9020 = .FALSE.
&if %:cur9020.AML$NEXT% = .TRUE. &then
&do
/*-ls9030
&do &until %eof9020%
/*-ls9040
&sv b = [write %output% ---]
/*
&type HECPREPRO:
&sv a = HECID:
&sv b = [write %output% %a%]
&sv c = %:cur9020.hecid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
```

```
/*attributes
&if ( %attrib% = 'yes' ) &then
&do
/*-ls9045
/*
&type HECPREPRO:
&sv a = X-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9020.hydronat//x-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = Y-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9020.hydronat//y-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = NODEZ:
&sv b = [write %output% %a%]
&sv c = %:cur9020.hydronat//nodez%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*-le9045
&end
/*
&type HECPREPRO:
&sv a = HECDNID:
&sv b = [write %output% %a%]
&sv c = %:cur9020.hydronat//hecdnid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*write to HMS basin file
/*
&if ( %hmsfile% = 'yes' ) &then
&do
/*ls-
/*
&type HECPREPRO:
&type HECPREPRO: Writing to HMS basin file
&type HECPREPRO:
/*
&sv a = 'Source: '
&sv b = %:cur9020.hecid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas X: '
&sv b = %:cur9020.hydronat//x-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas Y: '
&sv b = %:cur9020.hydronat//y-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Downstream: '
&sv b = %:cur9020.hydronat//hecdnid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = 'End:'
&sv b = [write %hmsbasin% %a%]
&sv a = ''
&sv b = [write %hmsbasin% %a%]
/*
```

```
/*-le
&end
/*
&r hecpause 3
cursor cur9020 next
&if %:cur9020.AML$NEXT% = .FALSE. &then
   &sv eof9020 = .TRUE.
/*-le9040
&end
/*-le9030
&end
cursor cur9020 remove
/*
/*--- reaches ---
/*
/*heading
/*
&sv b = [write %output% '---------------']
&sv b = [write %output% '--- REACHES ---']
&sv b = [write %output% '---------------']
&type HECPREPRO:
&type HECPREPRO: REACHES:
&type HECPREPRO:
/*
/*select reaches
/*
unselect symcov arcs
aselect symcov arcs ( hydroaat//hectype = 4 )
&if ( %.oblevel% ne 0 ) &then
&do
linecolor 3
arcs symcov
textcolor 7
arctext symcov hecid
&end
/*
/*write to general text file
/*
cursor cur9030 declare symcov arc ro
cursor cur9030 open
&sv eof9030 = .FALSE.
&if %:cur9030.AML$NEXT% = .TRUE. &then
&do
/*-ls9050
&do &until %eof9030%
/*-ls9060
&sv b = [write %output% ---]
/*
&type HECPREPRO:
&sv a = HECID:
&sv b = [write %output% %a%]
&sv c = %:cur9030.hecid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*attributes
&if ( %attrib% = 'yes' ) &then
&do
/*-ls9065
/*
&type HECPREPRO:
&sv a = HECLENGTH:
&sv b = [write %output% %a%]
&sv c = %:cur9030.hydroaat//heclength%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = HECSLOPE:
```

```
&sv b = [write %output% %a%]
&sv c = %:cur9030.hydroaat//hecslope%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*-le9065
&end
/*
&type HECPREPRO:
&sv a = HECUPID:
&sv b = [write %output% %a%]
&sv c = %:cur9030.hydroaat//hecupid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = HECDNID:
&sv b = [write %output% %a%]
&sv c = %:cur9030.hydroaat//hecdnid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*write to HMS basin file
/*
&if ( %hmsfile% = 'yes' ) &then
&do
/*ls-
/*
&type HECPREPRO:
&type HECPREPRO: Writing to HMS basin file
&type HECPREPRO:
/*
&sv a = 'Reach: '
&sv b = %:cur9030.hecid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Downstream: '
&sv b = %:cur9030.hydroaat//hecdnid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = 'End:'
&sv b = [write %hmsbasin% %a%]
&sv a = ''
&sv b = [write %hmsbasin% %a%]
/*
/*-le
&end
/*
&r hecpause 3
cursor cur9030 next
&if %:cur9030.AML$NEXT% = .FALSE. &then
   &sv eof9030 = .TRUE.
/*-le9060
&end
/*-le9050
&end
cursor cur9030 remove
/*
/*--- junctions ---
/*
/*heading
/*
&sv b = [ write %output% '-----------------' ]
&sv b = [ write %output% '--- JUNCTIONS ---' ]
&sv b = [ write %output% '-----------------' ]
&type HECPREPRO:
&type HECPREPRO: JUNCTIONS:
&type HECPREPRO:
```

```
/*
/*select junctions
/*
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 1 )
aselect symcov nodes ( hydronat//hectype = 3 )
aselect symcov nodes ( hydronat//hectype = 9 )
&if ( %.oblevel% ne 0 ) &then
&do
markersymbol 15
nodes symcov
textcolor 7
nodetext symcov hecid
&end
/*
/*write to general text file
/*
cursor cur9040 declare symcov node ro
cursor cur9040 open
&sv eof9040 = .FALSE.
&if %:cur9040.AML$NEXT% = .TRUE. &then
&do
/*-ls9070
&do &until %eof9040%
/*-ls9080
&sv b = [write %output% ---]
/*
&type HECPREPRO:
&sv a = HECID:
&sv b = [write %output% %a%]
&sv c = %:cur9040.hecid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*attributes
&if ( %attrib% = 'yes' ) &then
&do
/*-ls9085
&type HECPREPRO:
&sv a = X-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9040.hydronat//x-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = Y-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9040.hydronat//y-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = NODEZ:
&sv b = [write %output% %a%]
&sv c = %:cur9040.hydronat//nodez%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*-le9085
&end
/*
&type HECPREPRO:
&sv a = HECUPID(S):
&sv b = [write %output% %a%]
/*special processing for possible multiple hecupids
&sv h#tnodes = [ value hh#tnodes%:cur9040.hecid% ]
```

```
&if ( %h#tnodes% = 1 ) &then
&do
/*-ls9086
&sv c = [ value 1hecupid%:cur9040.hecid% ]
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*-le9086
&end
&if ( %h#tnodes% > 1 ) &then
&do
/*-ls9087
&sv i = 1
&sv j = %h#tnodes%
&do &while ( %i% <= %j% )
/*-ls9088
&sv matrix = %i%hecupid%:cur9040.hecid%
&sv c = [ value %matrix% ]
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&sv i = %i% + 1
/*-le9088
&end
&type HECPREPRO:
/*-le9087
&end
/*
&type HECPREPRO:
&sv a = HECDNID:
&sv b = [write %output% %a%]
&sv c = %:cur9040.hydronat//hecdnid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*write to HMS basin file
/*
&if ( %hmsfile% = 'yes' ) &then
&do
/*ls-
/*
&type HECPREPRO:
&type HECPREPRO: Writing to HMS basin file
&type HECPREPRO:
/*
&sv a = 'Junction: '
&sv b = %:cur9040.hecid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas X: '
&sv b = %:cur9040.hydronat//x-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas Y: '
&sv b = %:cur9040.hydronat//y-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Downstream: '
&sv b = %:cur9040.hydronat//hecdnid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = 'End:'
&sv b = [write %hmsbasin% %a%]
&sv a = ''
&sv b = [write %hmsbasin% %a%]
/*
/*-le
&end
/*
&r hecpause 3
cursor cur9040 next
```

```
&if %:cur9040.AML$NEXT% = .FALSE. &then
   &sv eof9040 = .TRUE.
/*-le9080
&end
/*-le9070
&end
cursor cur9040 remove
/*
/*--- reservoirs ---
/*
/*heading
/*
&sv b = [ write %output% '------------------' ]
&sv b = [ write %output% '--- RESERVOIRS ---' ]
&sv b = [ write %output% '------------------' ]
&type HECPREPRO:
&type HECPREPRO: RESERVOIRS:
&type HECPREPRO:
/*
/*select reservoirs
/*
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 10 )
&if ( %.oblevel% ne 0 ) &then
&do
markersymbol 15
nodes symcov
textcolor 7
nodetext symcov hecid
&end
/*
/*write to general text file
/*
cursor cur9050 declare symcov node ro
cursor cur9050 open
&sv eof9050 = .FALSE.
&if %:cur9050.AML$NEXT% = .TRUE. &then
&do
/*-ls9090
&do &until %eof9050%
/*-ls90100
&sv b = [write %output% ---]
/*
&type HECPREPRO:
&sv a = HECID:
&sv b = [write %output% %a%]
&sv c = %:cur9050.hecid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*attributes
&if ( %attrib% = 'yes' ) &then
&do
/*-ls90105
/*
&type HECPREPRO:
&sv a = X-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9050.hydronat//x-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = Y-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9050.hydronat//y-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
```

```
/*
&type HECPREPRO:
&sv a = NODEZ:
&sv b = [write %output% %a%]
&sv c = %:cur9050.hydronat//nodez%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*-le90105
&end
/*
&type HECPREPRO:
&sv a = HECUPID(S):
&sv b = [write %output% %a%]
/*special processing for possible multiple hecupids
&sv h#tnodes = [ value hh#tnodes%:cur9050.hecid% ]
&if ( %h#tnodes% = 1 ) &then
&do
/*-ls90107
&sv c = [ value 1hecupid%:cur9050.hecid% ]
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*-le90107
&end
&if ( %h#tnodes% > 1 ) &then
&do
/*-ls90108
&sv i = 1
&sv j = %h#tnodes%
&do &while ( %i% <= %j% )
/*-ls90109
&sv matrix = %i%hecupid%:cur9050.hecid%
&sv c = [ value %matrix% ]
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&sv i = %i% + 1
/*-le90109
&end
&type HECPREPRO:
/*-le90108
&end
/*
&type HECPREPRO:
&sv a = HECDNID(S):
&sv b = [write %output% %a%]
/*special processing for possible multiple hecdnids
&sv h#fnodes = [ value hh#fnodes%:cur9050.hecid% ]
&if ( %h#fnodes% = 1 ) &then
&do
/*-ls90109.2
&sv c = %:cur9050.hydronat//hecdnid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*-le90109.2
&end
&if ( %h#fnodes% > 1 ) &then
&do
/*-ls90109.4
&sv i = 1
&sv j = %h#fnodes%
&do &while ( %i% <= %j% )
/*-ls90109.6
&sv matrix = %i%hecdnid%:cur9050.hecid%
&sv c = [ value %matrix% ]
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&sv i = %i% + 1
/*-le90109.6
```

```
&end
&type HECPREPRO:
/*-le90109.4
&end
/*
/*write to HMS basin file
/*
&if ( %hmsfile% = 'yes' ) &then
&do
/*ls-
/*
&type HECPREPRO:
&type HECPREPRO: Writing to HMS basin file
&type HECPREPRO:
/*
&sv a = 'Reservoir: '
&sv b = %:cur9050.hecid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas X: '
&sv b = %:cur9050.hydronat//x-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas Y: '
&sv b = %:cur9050.hydronat//y-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Downstream: '
/*special processing for possible multiple hecdnids
&sv h#fnodes = [ value hh#fnodes%:cur9050.hecid% ]
&if ( %h#fnodes% = 1 ) &then
&do
/*-ls90109.2b
&sv b = %:cur9050.hydronat//hecdnid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*-le90109.2b
&end
&if ( %h#fnodes% > 1 ) &then
&do
/*-ls90109.4b
&sv b =
&sv i = 1
&sv j = %h#fnodes%
&do &while ( %i% <= %j% )
/*-ls90109.6b
&sv matrix = %i%hecdnid%:cur9050.hecid%
&sv b = %b% [ value %matrix% ]
&type HECPREPRO: %a% %c%
&sv i = %i% + 1
/*-le90109.6b
&end
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*-le90109.4b
&end
/*
&sv a = 'End:'
&sv b = [write %hmsbasin% %a%]
&sv a = ''
&sv b = [write %hmsbasin% %a%]
/*
/*-le
&end
/*
&r hecpause 3
cursor cur9050 next
&if %:cur9050.AML$NEXT% = .FALSE. &then
   &sv eof9050 = .TRUE.
/*-le90100
&end
/*-le9090
&end
```

```
cursor cur9050 remove
/*
/*--- diversions ---
/*
/*heading
/*
&sv b = [ write %output% '------------------' ]
&sv b = [ write %output% '--- DIVERSIONS ---' ]
&sv b = [ write %output% '------------------' ]
&type HECPREPRO:
&type HECPREPRO: DIVERSIONS:
&type HECPREPRO:
/*
/*select diversions
/*
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 8 )
&if ( %.oblevel% ne 0 ) &then
&do
markersymbol 11
nodes symcov
textcolor 7
nodetext symcov hecid
&end
/*
/*write to general text file
/*
cursor cur9060 declare symcov node ro
cursor cur9060 open
&sv eof9060 = .FALSE.
&if %:cur9060.AML$NEXT% = .TRUE. &then
&do
/*-ls90110
&do &until %eof9060%
/*-ls90120
&sv b = [write %output% ---]
/*
&type HECPREPRO:
&sv a = HECID:
&sv b = [write %output% %a%]
&sv c = %:cur9060.hecid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*attributes
&if ( %attrib% = 'yes' ) &then
&do
/*-ls90125
/*
&type HECPREPRO:
&sv a = X-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9060.hydronat//x-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = Y-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9060.hydronat//y-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = NODEZ:
&sv b = [write %output% %a%]
&sv c = %:cur9060.hydronat//nodez%
&sv d = [write %output% %c%]
```

```
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*-le90125
&end
/*
&type HECPREPRO:
&sv a = HECUPID:
&sv b = [write %output% %a%]
&sv c = [ value 1hecupid%:cur9060.hecid% ]
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = HECDNIDS:
&sv b = [write %output% %a%]
/*special for diversions
&sv h#fnodes = [ value hh#fnodes%:cur9060.hecid% ]
&sv i = 1
&sv j = %h#fnodes%
&do &while ( %i% <= %j% )
/*-ls90126
&sv matrix = %i%hecdnid%:cur9060.hecid%
&sv c = [ value %matrix% ]
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&sv i = %i% + 1
/*-le90126
&end
&type HECPREPRO:
/*
/*write to HMS basin file
/*
&if ( %hmsfile% = 'yes' ) &then
&do
/*ls-
/*
&type HECPREPRO:
&type HECPREPRO: Writing to HMS basin file
&type HECPREPRO:
/*
&sv a = 'Diversion: '
&sv b = %:cur9060.hecid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas X: '
&sv b = %:cur9060.hydronat//x-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas Y: '
&sv b = %:cur9060.hydronat//y-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Downstream: '
/*special for diversions
&sv h#fnodes = [ value hh#fnodes%:cur9060.hecid% ]
&sv b =
&sv i = 1
&sv j = %h#fnodes%
&do &while ( %i% <= %j% )
/*-ls90126b
&sv matrix = %i%hecdnid%:cur9060.hecid%
&sv b = %b% [ value %matrix% ]
&sv i = %i% + 1
/*-le90126bb
&end
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = 'End:'
&sv b = [write %hmsbasin% %a%]
```

```
&sv a = ''
&sv b = [write %hmsbasin% %a%]
/*
/*-le
&end
/*
&r hecpause 3
cursor cur9060 next
&if %:cur9060.AML$NEXT% = .FALSE. &then
   &sv eof9060 = .TRUE.
/*-le90120
&end
/*-le90110
&end
cursor cur9060 remove
/*
/*--- sinks ---
/*
/*heading
/*
&sv b = [ write %output% '-------------' ]
&sv b = [ write %output% '--- SINKS ---' ]
&sv b = [ write %output% '-------------' ]
&type HECPREPRO:
&type HECPREPRO: SINKS:
&type HECPREPRO:
/*
/*select sinks
/*
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 6 )
&if ( %.oblevel% ne 0 ) &then
&do
markersymbol 23
nodes symcov
textcolor 7
nodetext symcov hecid
&end
/*
/*write to general text file
/*
cursor cur9070 declare symcov node ro
cursor cur9070 open
&sv eof9070 = .FALSE.
&if %:cur9070.AML$NEXT% = .TRUE. &then
&do
/*-ls90130
&do &until %eof9070%
/*-ls90140
&sv b = [write %output% ---]
/*
&type HECPREPRO:
&sv a = HECID:
&sv b = [write %output% %a%]
&sv c = %:cur9070.hecid%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*attributes
&if ( %attrib% = 'yes' ) &then
&do
/*-ls90145
/*
&type HECPREPRO:
&sv a = X-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9070.hydronat//x-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
```

```
/*
&type HECPREPRO:
&sv a = Y-COORD:
&sv b = [write %output% %a%]
&sv c = %:cur9070.hydronat//y-coord%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
&type HECPREPRO:
&sv a = NODEZ:
&sv b = [write %output% %a%]
&sv c = %:cur9070.hydronat//nodez%
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*
/*-le90145
&end
/*
&type HECPREPRO:
&sv a = HECUPID(S):
&sv b = [write %output% %a%]
/*special processing for possible multiple hecupids
&sv h#tnodes = [ value hh#tnodes%:cur9070.hecid% ]
&if ( %h#tnodes% = 1 ) &then
&do
/*-ls90146
&sv c = [ value 1hecupid%:cur9070.hecid% ]
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&type HECPREPRO:
/*-le90146
&end
&if ( %h#tnodes% > 1 ) &then
&do
/*-ls90147
&sv i = 1
&sv j = %h#tnodes%
&do &while ( %i% <= %j% )
/*-ls90148
&sv matrix = %i%hecupid%:cur9070.hecid%
&sv c = [ value %matrix% ]
&sv d = [write %output% %c%]
&type HECPREPRO: %a% %c%
&sv i = %i% + 1
/*-le90148
&end
&type HECPREPRO:
/*-le90147
&end
/*
/*write to HMS basin file
/*
&if ( %hmsfile% = 'yes' ) &then
&do
/*ls-
/*
&type HECPREPRO:
&type HECPREPRO: Writing to HMS basin file
&type HECPREPRO:
/*
&sv a = 'Sink: '
&sv b = %:cur9070.hecid%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas X: '
&sv b = %:cur9070.hydronat//x-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = '     Canvas Y: '
```

```
&sv b = %:cur9070.hydronat//y-coord%
&sv c = [write %hmsbasin% [quote [unquote %a%] %b%]]
/*
&sv a = 'End:'
&sv b = [write %hmsbasin% %a%]
&sv a = ''
&sv b = [write %hmsbasin% %a%]
/*
/*-le
&end
/*
&r hecpause 3
cursor cur9070 next
&if %:cur9070.AML$NEXT% = .FALSE. &then
   &sv eof9070 = .TRUE.
/*-le90140
&end
/*-le90130
&end
cursor cur9070 remove
/*
/*--- close general output file ---
/*
&sv b = [write %output% '-----------']
&sv b = [write %output% '--- END ---']
&sv b = [write %output% '-----------']
&sv a = [close %output%]
/*
/*--- close HMS basin file ---
/*
&sv a = [close %hmsbasin%]
/*
/*---------------------------
/*--- Clean Up Some Files ---
/*---------------------------
/*
&sys arc kill streamcov all
/*
/*-------------------------
/*--- User Examination ---
/*-------------------------
/*
&type
&type HECPREPRO:
&type HECPREPRO: SYMCOV LEGEND:
&type HECPREPRO: ----------------------------------------
&type HECPREPRO:
&type HECPREPRO: LINES:
&type HECPREPRO: White:  Input Stream and Subbasin Lines
&type HECPREPRO: Green:  Reaches
&type HECPREPRO: Yellow: Links
&type HECPREPRO:
&type HECPREPRO: SYMBOLS:
&type HECPREPRO: Squares:              Subbasins
&type HECPREPRO: Stars:                Sources
&type HECPREPRO: Triangles:            Junctions
&type HECPREPRO: Upsidedown Triangles: Reservoirs
&type HECPREPRO: Circles:              Diversions
&type HECPREPRO: Diamonds:             Sinks
&type HECPREPRO:
&type HECPREPRO: TEXT:
&type HECPREPRO: Yellow: HECID of Elements
&type HECPREPRO:
&type HECPREPRO: ----------------------------------------
&type HECPREPRO:
&type HECPREPRO: General Text File: %outfile%
&if ( %dxffile% = 'yes' ) &then
&type HECPREPRO: DXF File:          %dxfname%
&if ( %hmsfile% = 'yes' ) &then
&type HECPREPRO: HMS Basin File:    %hmsname%
&type HECPREPRO:
```

```
&type HECPREPRO: ----------------------------------------
&type HECPREPRO:
&type
&r hecpause 2
/*
/*----------------
/*----------------
/*--- Close Up ---
/*----------------
/*----------------
/*
&type HECPREPRO:
&type HECPREPRO: NORMAL END
&type HECPREPRO: ERRORS DETECTED: %errors%
&type HECPREPRO:
&type HECPREPRO: ---------------------
&type HECPREPRO: --- HECPREPRO END ---
&type HECPREPRO: ---------------------
&type HECPREPRO:
/*
&messages &on
/*
&return
/*
/*-----------
/*-----------
/*--- End ---
/*-----------
/*-----------
/*
```

A.2. Pause Utility (hecpause.aml).

```
/*---
/*Name: hecpause.aml
/*Date: 06/10/96
/*Author: Ferdinand Hellweger, UT Austin
/*---
/*Purpose/Description:
/*This AML pauses if the observation level exceeds the observation
/*threshold communicated over the command line.
/*---
/*Called by:
/*hecprepro.aml
/*---
&args ob
&if ( %.oblevel% >= %ob% ) &then
&do
&sv .oblevel [ response 'HECPREPRO:<Return> or new observation le~
vel to continue' %.oblevel% ]
&if ( %.oblevel% = 9 ) &then
   &stop HECPREPRO: PREMATURE END
&end
&return
```

A.3. Shell Program (hecshell.aml).

```
/*---
/*Name: hecshell.aml
```

```
/*Date: 06/10/96
/*Author: Ferdinand Hellweger, UT Austin
/*---
/*Purpose/Description:
/*This AML is a shell that prompts the user for the necessary data and
/*then runs hecprepro.aml.
/*---
/*
&type HECPREPRO:
&type HECPREPRO: ------------------------
&type HECPREPRO: ------------------------
&type HECPREPRO: --- HECPREPRO SHELL ---
&type HECPREPRO: ------------------------
&type HECPREPRO: ------------------------
&type HECPREPRO:
/*
&if [exists hecprev.txt] &then
&do
&sv prevrun = [open hecprev.txt status -read]
&sv oldstreamcov = [read %prevrun% status]
&sv oldsubcov = [read %prevrun% status]
&sv oldelevgrid = [read %prevrun% status]
&sv .oldoblevel = [read %prevrun% status]
&sv oldattrib = [read %prevrun% status]
&sv oldclipped = [read %prevrun% status]
&sv oldsnapit = [read %prevrun% status]
&sv oldsnaptol = [read %prevrun% status]
&sv oldoutfile = [read %prevrun% status]
&sv olddxffile = [read %prevrun% status]
&sv olddxfname = [read %prevrun% status]
&sv oldhmsfile = [read %prevrun% status]
&sv oldhmsname = [read %prevrun% status]
&sv a = [close %prevrun%]
&label e2
&type HECPREPRO:
&type HECPREPRO: ---------------------------------
&type HECPREPRO: --- Same Data as Previous Run? ---
&type HECPREPRO: ---------------------------------
&type HECPREPRO:
&type HECPREPRO: Description:
&type HECPREPRO: The program can be run with the same input as the
&type HECPREPRO: previous run.  The previous input variables were:
&type HECPREPRO:
&type HECPREPRO: Stream Coverage: %oldstreamcov%
&type HECPREPRO: Subbasin Coverage: %oldsubcov%
&type HECPREPRO: Elevation Grid: %oldelevgrid%
&type HECPREPRO: User Observation Level: %.oldoblevel%
&type HECPREPRO: Attribute Collection? %oldattrib%
&type HECPREPRO: Clipped? %oldclipped%
&type HECPREPRO: Node Snapping? %oldsnapit%
&type HECPREPRO: Snapping Tolerance Distance: %oldsnaptol%
&type HECPREPRO: Name of General Text File: %oldoutfile%
&type HECPREPRO: DXF File? %olddxffile%
&type HECPREPRO: Name of DXF File: %olddxfname%
&type HECPREPRO: HMS Basin File? %oldhmsfile%
&type HECPREPRO: Name of HMS Basin File: %oldhmsname%
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Y - same data will be used
&type HECPREPRO: N - same data will not be used (default)
&type HECPREPRO:
&sv sameprev [ response 'HECPREPRO: Same Data as Previous Run?']
&if ( [ length %sameprev% ] = 0 ) &then
&do
&sv sameprev = no
&type HECPREPRO:
&type HECPREPRO: Same data will not be used (default)
&type HECPREPRO:
&end
&if ( %sameprev% = 'y' ) &then
   &sv sameprev = 'yes'
```

```
&if ( %sameprev% = 'Y' ) &then
   &sv sameprev = 'yes'
&if ( %sameprev% = 'YES' ) &then
   &sv sameprev = 'yes'
&if ( %sameprev% = 'n' ) &then
   &sv sameprev = 'no'
&if ( %sameprev% = 'N' ) &then
   &sv sameprev = 'no'
&if ( %sameprev% = 'NO' ) &then
   &sv sameprev = 'no'
&sv error = yes
&if ( %sameprev% = 'yes' ) &then
   &sv error = no
&if ( %sameprev% = 'no' ) &then
   &sv error = no
&if ( %error% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR
&type HECPREPRO:
&goto e2
&end
&if ( %sameprev% = 'yes' ) &then
&do
&sv streamcov = %oldstreamcov%
&sv subcov = %oldsubcov%
&sv elevgrid = %oldelevgrid%
&sv .oblevel = %.oldoblevel%
&sv attrib = %oldattrib%
&sv clipped = %oldclipped%
&sv snapit = %oldsnapit%
&sv snaptol = %oldsnaptol%
&sv outfile = %oldoutfile%
&sv dxffile = %olddxffile%
&sv dxfname = %olddxfname%
&sv hmsfile = %oldhmsfile%
&sv hmsname = %oldhmsname%
&goto z
&end
&end
&if ( not [exists hecprev.txt] ) &then
&do
&sv oldstreamcov = tkst1
&sv oldsubcov = tksub
&sv oldelevgrid = tkelev
&sv .oldoblevel = 2
&sv oldattrib = yes
&sv oldclipped = no
&sv oldsnapit = no
&sv oldsnaptol = 0
&sv oldoutfile = hecprepro.out
&sv olddxffile = no
&sv olddxfname = dxfout.dxf
&sv oldhmsfile = no
&sv oldhmsname = hmsbasin.txt
&end
/*
&label a
&type HECPREPRO:
&type HECPREPRO: ------------------------
&type HECPREPRO: --- Stream Coverage ---
&type HECPREPRO: ------------------------
&type HECPREPRO:
&type HECPREPRO: Description:
&type HECPREPRO: The stream coverage has to have all the arcs pointing
&type HECPREPRO: downstream.  Polygons are considered reservoirs.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: The name of a stream coverage
&type HECPREPRO:
&sv streamcov [ response 'HECPREPRO: Stream Coverage <'%oldstreamcov%'>']
```

```
&if ( [ length %streamcov% ] = 0 ) &then
&sv streamcov = %oldstreamcov%
&if ( not [ exists %streamcov% -cover ] ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR - Coverage does not exist
&type HECPREPRO:
&goto a
&end
/*
&label b
&type HECPREPRO:
&type HECPREPRO: -------------------------
&type HECPREPRO: --- Subbasin Coverage ---
&type HECPREPRO: -------------------------
&type HECPREPRO:
&type HECPREPRO: Description:
&type HECPREPRO: Each subbasin has to contain an arc from
&type HECPREPRO: the stream coverage.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: The name of a subbasin coverage
&type HECPREPRO:
&sv subcov [ response 'HECPREPRO: Subbasin Coverage <'%oldsubcov%'>']
&if ( [ length %subcov% ] = 0 ) &then
&sv subcov = %oldsubcov%
&if ( not [ exists %subcov% -cover ] ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR - Coverage does not exist
&type HECPREPRO:
&goto b
&end
/*
&label c
&type HECPREPRO:
&type HECPREPRO: ----------------------
&type HECPREPRO: --- Elevation Grid ---
&type HECPREPRO: ----------------------
&type HECPREPRO:
&type HECPREPRO: Description:
&type HECPREPRO: The grid has to cover every node in the
&type HECPREPRO: stream coverage.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: The name of an elevation grid
&type HECPREPRO:
&sv elevgrid [ response 'HECPREPRO: Elevation Grid <'%oldelevgrid%'>']
&if ( [ length %elevgrid% ] = 0 ) &then
&sv elevgrid = %oldelevgrid%
&if ( not [ exists %elevgrid% -grid ] ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR - Grid does not exist
&type HECPREPRO:
&goto c
&end
/*
&label d
&type HECPREPRO:
&type HECPREPRO: ------------------------------
&type HECPREPRO: --- User Observation Level ---
&type HECPREPRO: ------------------------------
&type HECPREPRO:
&type HECPREPRO: Description:
&type HECPREPRO: The user observation level controls how often the program
&type HECPREPRO: pauses.  At each pause the user can change to another user
&type HECPREPRO: observation level by typing it in or quit instantly by typing
&type HECPREPRO: in 9.  One can also 'jump' to a section in the program by
&type HECPREPRO: typing in the fraction of the section.
&type HECPREPRO:
```

```
&type HECPREPRO: Valid Responses:
&type HECPREPRO: 0 - no pause and no graphic display (for background running)
&type HECPREPRO: 1 - pause at error
&type HECPREPRO: 2 - pause at error and legends
&type HECPREPRO: 3 - first level debug
&type HECPREPRO: 4 - second level debug
&type HECPREPRO: 0.10, 0.20, etc. - makes first pause at STEP 10, STEP 20,
etc.
&type HECPREPRO:
&sv .oblevel [ response 'HECPREPRO: User Observation Level <'%.oldoblevel%'>']
&if ( [ length %.oblevel% ] = 0 ) &then
&sv .oblevel = %.oldoblevel%
&sv error = yes
&if ( %.oblevel% = 0 ) &then
   &sv error = no
&if ( %.oblevel% = 1 ) &then
   &sv error = no
&if ( %.oblevel% = 2 ) &then
   &sv error = no
&if ( %.oblevel% = 3 ) &then
   &sv error = no
&if ( %.oblevel% = 4 ) &then
   &sv error = no
&if ( %.oblevel% = 0.10 ) &then
   &sv error = no
&if ( %.oblevel% = 0.20 ) &then
   &sv error = no
&if ( %.oblevel% = 0.30 ) &then
   &sv error = no
&if ( %.oblevel% = 0.40 ) &then
   &sv error = no
&if ( %.oblevel% = 0.50 ) &then
   &sv error = no
&if ( %.oblevel% = 0.60 ) &then
   &sv error = no
&if ( %.oblevel% = 0.70 ) &then
   &sv error = no
&if ( %.oblevel% = 0.80 ) &then
   &sv error = no
&if ( %.oblevel% = 0.90 ) &then
   &sv error = no
&if ( %error% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR
&type HECPREPRO:
&goto d
&end
/*
&label e
&type HECPREPRO:
&type HECPREPRO: -----------------------------
&type HECPREPRO: --- Attribute Collection? ---
&type HECPREPRO: -----------------------------
&type HECPREPRO:
&type HECPREPRO: Description:
&type HECPREPRO: Collecting attributes is optional.  If attribute are not
&type HECPREPRO: collected the program will only establish connectivity
&type HECPREPRO: of the elements.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Y - attributes will be collected
&type HECPREPRO: N - attributes will not be collected
&type HECPREPRO:
&sv attrib [ response 'HECPREPRO: Attribute Collection? <'%oldattrib%'>']
&if ( [ length %attrib% ] = 0 ) &then
&sv attrib = %oldattrib%
&if ( %attrib% = 'y' ) &then
   &sv attrib = 'yes'
&if ( %attrib% = 'Y' ) &then
   &sv attrib = 'yes'
```

```
&if ( %attrib% = 'YES' ) &then
   &sv attrib = 'yes'
&if ( %attrib% = 'n' ) &then
   &sv attrib = 'no'
&if ( %attrib% = 'N' ) &then
   &sv attrib = 'no'
&if ( %attrib% = 'NO' ) &then
   &sv attrib = 'no'
&sv error = yes
&if ( %attrib% = 'yes' ) &then
   &sv error = no
&if ( %attrib% = 'no' ) &then
   &sv error = no
&if ( %error% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR
&type HECPREPRO:
&goto e
&end
/*
&label f
&type HECPREPRO:
&type HECPREPRO: ----------------
&type HECPREPRO: --- Clipped? ---
&type HECPREPRO: ----------------
&type HECPREPRO:
&type HECPREPRO: Description:
&type HECPREPRO: Was the stream coverage clipped or intersected with
&type HECPREPRO: the subbasin coverage?  Choose 'Y' if you are unsure.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Y - the stream coverage was clipped
&type HECPREPRO: N - the stream coverage was not clipped
&type HECPREPRO:
&sv clipped [ response 'HECPREPRO: Clipped? <'%oldclipped%'>']
&if ( [ length %clipped% ] = 0 ) &then
&sv clipped = %oldclipped%
&if ( %clipped% = 'y' ) &then
   &sv clipped = 'yes'
&if ( %clipped% = 'Y' ) &then
   &sv clipped = 'yes'
&if ( %clipped% = 'YES' ) &then
   &sv clipped = 'yes'
&if ( %clipped% = 'n' ) &then
   &sv clipped = 'no'
&if ( %clipped% = 'N' ) &then
   &sv clipped = 'no'
&if ( %clipped% = 'NO' ) &then
   &sv clipped = 'no'
&sv error = yes
&if ( %clipped% = 'yes' ) &then
   &sv error = no
&if ( %clipped% = 'no' ) &then
   &sv error = no
&if ( %error% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR
&type HECPREPRO:
&goto f
&end
/*
&label g
&type HECPREPRO:
&type HECPREPRO: ----------------------
&type HECPREPRO: --- Node Snapping? ---
&type HECPREPRO: ----------------------
&type HECPREPRO:
&type HECPREPRO: Description
&type HECPREPRO: Nodes from the stream coverage can be snapped to nodes
```

```
&type HECPREPRO: from the subbasin coverage.  This is useful for data
&type HECPREPRO: created with GRID routines.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Y - nodes will be snapped
&type HECPREPRO: N - nodes will not be snapped
&type HECPREPRO:
&sv snapit [ response 'HECPREPRO: Node Snapping? <'%oldsnapit%'>']
&if ( [ length %snapit% ] = 0 ) &then
&sv snapit = %oldsnapit%
&if ( %snapit% = 'y' ) &then
   &sv snapit = 'yes'
&if ( %snapit% = 'Y' ) &then
   &sv snapit = 'yes'
&if ( %snapit% = 'YES' ) &then
   &sv snapit = 'yes'
&if ( %snapit% = 'n' ) &then
   &sv snapit = 'no'
&if ( %snapit% = 'N' ) &then
   &sv snapit = 'no'
&if ( %snapit% = 'NO' ) &then
   &sv snapit = 'no'
&sv error = yes
&if ( %snapit% = 'yes' ) &then
   &sv error = no
&if ( %snapit% = 'no' ) &then
   &sv error = no
&if ( %error% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR
&type HECPREPRO:
&goto g
&end
/*
&label h
&if ( %snapit% = 'no' ) &then
   &sv snaptol = 0
&if ( %snapit% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: ------------------------------------
&type HECPREPRO: --- Snapping Tolerance Distance ---
&type HECPREPRO: ------------------------------------
&type HECPREPRO:
&type HECPREPRO: Description
&type HECPREPRO: Only nodes from the stream coverage within this distance
&type HECPREPRO: of nodes from the subbasin coverage will be snapped.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Any number
&type HECPREPRO:
&sv snaptol [ response 'HECPREPRO: Snapping Tolerance Distance
<'%oldsnaptol%'>']
&if ( [ length %snaptol% ] = 0 ) &then
&sv snaptol = %oldsnaptol%
&end
/*
&type HECPREPRO:
&type HECPREPRO: ---------------------------------
&type HECPREPRO: --- Name of General Text File ---
&type HECPREPRO: ---------------------------------
&type HECPREPRO:
&type HECPREPRO: Description
&type HECPREPRO: The general feature summary will be written to this file.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Any valid file name.
&type HECPREPRO:
&type HECPREPRO:
```

```
&sv outfile [ response 'HECPREPRO: Name of General Text File
<'%oldoutfile%'>']
&if ( [ length %outfile% ] = 0 ) &then
&sv outfile = %oldoutfile%
/*
&label i
&type HECPREPRO:
&type HECPREPRO: -----------------
&type HECPREPRO: --- DXF File? ---
&type HECPREPRO: -----------------
&type HECPREPRO:
&type HECPREPRO: Description
&type HECPREPRO: An AutoCAD Drawing Exchange File can be created.
&type HECPREPRO: The file would contain two layers, one for the
&type HECPREPRO: streams and one for the subbasin boundaries.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Y - DXF file will be created.
&type HECPREPRO: N - DXF file will not be created.
&type HECPREPRO:
&sv dxffile [ response 'HECPREPRO: DXF File? <'%olddxffile%'>']
&if ( [ length %dxffile% ] = 0 ) &then
&sv dxffile = %olddxffile%
&if ( %dxffile% = 'y' ) &then
   &sv dxffile = 'yes'
&if ( %dxffile% = 'Y' ) &then
   &sv dxffile = 'yes'
&if ( %dxffile% = 'YES' ) &then
   &sv dxffile = 'yes'
&if ( %dxffile% = 'n' ) &then
   &sv dxffile = 'no'
&if ( %dxffile% = 'N' ) &then
   &sv dxffile = 'no'
&if ( %dxffile% = 'NO' ) &then
   &sv dxffile = 'no'
&sv error = yes
&if ( %dxffile% = 'yes' ) &then
   &sv error = no
&if ( %dxffile% = 'no' ) &then
   &sv error = no
&if ( %error% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR
&type HECPREPRO:
&goto i
&end
/*
&if ( %dxffile% = 'no' ) &then
   &sv dxfname = dxfout.dxf
&if ( %dxffile% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: ------------------------
&type HECPREPRO: --- Name of DXF File ---
&type HECPREPRO: ------------------------
&type HECPREPRO:
&type HECPREPRO: Description
&type HECPREPRO: The DXF data will be written to this file.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Any valid file name.
&type HECPREPRO:
&sv dxfname [ response 'HECPREPRO: Name of DXF File <'%olddxfname%'>']
&if ( [ length %dxfname% ] = 0 ) &then
&sv outfile = %olddxfname%
&end
/*
&label j
&type HECPREPRO:
&type HECPREPRO: ------------------------
```

```
&type HECPREPRO: --- HMS Basin File? ---
&type HECPREPRO: ----------------------
&type HECPREPRO:
&type HECPREPRO: Description
&type HECPREPRO: A HMS Basin File can be created.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Y - HMS basin file will be created.
&type HECPREPRO: N - HMS basin file will not be created.
&type HECPREPRO:
&sv hmsfile [ response 'HECPREPRO: HMS Basin File? <'%oldhmsfile%'>']
&if ( [ length %hmsfile% ] = 0 ) &then
&sv hmsfile = %oldhmsfile%
&if ( %hmsfile% = 'y' ) &then
   &sv hmsfile = 'yes'
&if ( %hmsfile% = 'Y' ) &then
   &sv hmsfile = 'yes'
&if ( %hmsfile% = 'YES' ) &then
   &sv hmsfile = 'yes'
&if ( %hmsfile% = 'n' ) &then
   &sv hmsfile = 'no'
&if ( %hmsfile% = 'N' ) &then
   &sv hmsfile = 'no'
&if ( %hmsfile% = 'NO' ) &then
   &sv hmsfile = 'no'
&sv error = yes
&if ( %hmsfile% = 'yes' ) &then
   &sv error = no
&if ( %hmsfile% = 'no' ) &then
   &sv error = no
&if ( %error% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: INPUT ERROR
&type HECPREPRO:
&goto j
&end
/*
&if ( %hmsfile% = 'no' ) &then
   &sv hmsname = hmsbasin.txt
&if ( %hmsfile% = 'yes' ) &then
&do
&type HECPREPRO:
&type HECPREPRO: ------------------------------
&type HECPREPRO: --- Name of HMS Basin File ---
&type HECPREPRO: ------------------------------
&type HECPREPRO:
&type HECPREPRO: Description
&type HECPREPRO: The HMS basin data will be written to this file.
&type HECPREPRO:
&type HECPREPRO: Valid Responses:
&type HECPREPRO: Any valid file name.
&type HECPREPRO:
&sv hmsname [ response 'HECPREPRO: Name of HMS Basin File <'%oldhmsname%'>']
&if ( [ length %hmsname% ] = 0 ) &then
&sv hmsname = %oldhmsname%
&end
/*
&label z
&sv prevrun = [open hecprev.txt status -write]
&sv a = [write %prevrun% %streamcov%]
&sv a = [write %prevrun% %subcov%]
&sv a = [write %prevrun% %elevgrid%]
&sv a = [write %prevrun% %.oblevel%]
&sv a = [write %prevrun% %attrib%]
&sv a = [write %prevrun% %clipped%]
&sv a = [write %prevrun% %snapit%]
&sv a = [write %prevrun% %snaptol%]
&sv a = [write %prevrun% %outfile%]
&sv a = [write %prevrun% %dxffile%]
&sv a = [write %prevrun% %dxfname%]
```

```
&sv a = [write %prevrun% %hmsfile%]
&sv a = [write %prevrun% %hmsname%]
&sv a = [close %prevrun%]
&type HECPREPRO:
&type HECPREPRO: Starting HECPREPRO ...
&type HECPREPRO:
/*
&r hecprepr %streamcov% %subcov% %elevgrid% %.oblevel% %attrib% %clipped% ~
%snapit% %snaptol% %outfile% %dxffile% %dxfname% %hmsfile% %hmsname%
/*
&return
```

A.4. Hydrocov Display Program (hechydro.aml).

```
/*---
/*Name: hechydro.aml
/*Date: 06/10/96
/*Author: Ferdinand Hellweger, UT Austin
/*---
/*Purpose/Description:
/*This AML reproduces the hydrocov display
/*---
/*
clear
/*
/*streamcov and subcov
linecolor 1
markersymbol 17
arcs streamcov
nodes streamcov
arcs subcov
/*channel system
unselect hydrocov arcs
unselect hydrocov nodes
aselect hydrocov arc ( hectype = 4 )
aselect hydrocov node ( hectype = 4 )
linecolor 7
arcs hydrocov
nselect hydrocov node
nselect hydrocov arc
unselect hydrocov node ( hectype = 1 )
linecolor 2
arcs hydrocov
unselect hydrocov poly
aselect hydrocov poly
unselect hydrocov poly ( hydrocov# = 1 )
linecolor 5
polygons hydrocov
/*subbasin outlet
unselect hydrocov nodes
aselect hydrocov nodes ( hectype = 1 )
markersymbol 15
nodes hydrocov
/*junction
unselect hydrocov nodes
aselect hydrocov nodes ( hectype = 3 )
markersymbol 11
nodes hydrocov
/*diversion
unselect hydrocov nodes
aselect hydrocov nodes ( hectype = 8 )
markersymbol 12
nodes hydrocov
/*sink
unselect hydrocov nodes
aselect hydrocov nodes ( hectype = 6 )
markersymbol 14
nodes hydrocov
```

```
/*source
unselect hydrocov nodes
aselect hydrocov nodes ( hectype = 7 )
markersymbol 16
nodes hydrocov
/*reservoir
unselect hydrocov nodes
aselect hydrocov nodes ( hectype = 10 )
markersymbol 42
nodes hydrocov
/*subbasin
aselect subcov poly
markersymbol 7
points subcov
/*channel elements text
textcolor 1
unselect hydrocov nodes
aselect hydrocov nodes ( hectype ne 2 )
unselect hydrocov nodes ( hectype = 4 )
nodetext hydrocov hecid
textcolor 5
nodetext hydrocov hecdnid ur
/*channel text
unselect hydrocov arcs
aselect hydrocov arcs ( hectype ne 2 )
arctext hydrocov hecdnid ur
textcolor 7
arctext hydrocov hecid
/*subbasin text
textcolor 3
pointtext subcov hecid
textcolor 5
pointtext subcov hecdnid ur

/*
&type
&type HECPREPRO:
&type HECPREPRO: HYDROCOV LEGEND:
&type HECPREPRO: ------------------------------------------------
&type HECPREPRO:
&type HECPREPRO: LINES:
&type HECPREPRO: White:  Subbasin Boundary and Streams Outside
&type HECPREPRO:         Watershed Boundary
&type HECPREPRO: Yellow: HEC Channel Network
&type HECPREPRO: Red:    Non HEC Channel Network
&type HECPREPRO: Cyan:   Reservoir Shoreline
&type HECPREPRO:
&type HECPREPRO: SYMBOLS:
&type HECPREPRO: Green Triangles:           Subbasin Outlets
&type HECPREPRO: Red Triangles:             Sinks
&type HECPREPRO: Blue Triangles:            Sources
&type HECPREPRO: Green Circles:             Junctions
&type HECPREPRO: Blue Circles:              Diversions
&type HECPREPRO: Green Boxes:               Subbasins
&type HECPREPRO: Upsidedown Green Triangle: Reservoirs
&type HECPREPRO:
&type HECPREPRO: TEXT:
&type HECPREPRO: White:  HECID of Channel Elements
&type HECPREPRO: Yellow: HECID of Reaches
&type HECPREPRO: Cyan:   HECDNID of Elements
&type HECPREPRO:         (diversions show zero)
&type HECPREPRO:
&type HECPREPRO: ------------------------------------------------
&type HECPREPRO:
&type
/*
&return
```

A.5. Symcov Display Program (hecsym.aml).

```
/*---
/*Name: hecsym.aml
/*Date: 06/10/96
/*Author: Ferdinand Hellweger, UT Austin
/*---
/*Purpose/Description:
/*This AML reproduces the symcov display
/*---
/*
clear
/*
/*streamcov and subcov
linecolor 1
arcs streamcov
arcs subcov
/*
/*symcov
aselect symcov arcs
linecolor 7
arcs symcov
/*subbasin
unselect symcov nodes
aselect symcov nodes ( subpat//hectype = 5 )
markersymbol 7
nodes symcov
textcolor 7
nodetext symcov hecid
/*source
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 7 )
markersymbol 47
nodes symcov
textcolor 7
nodetext symcov hecid
/*reach
unselect symcov arcs
aselect symcov arcs ( hydroaat//hectype = 4 )
linecolor 3
arcs symcov
textcolor 7
arctext symcov hecid
/*junction
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 1 )
aselect symcov nodes ( hydronat//hectype = 3 )
aselect symcov nodes ( hydronat//hectype = 9 )
markersymbol 15
nodes symcov
textcolor 7
nodetext symcov hecid
/*reservoir
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 10 )
markersymbol 42
nodes symcov
textcolor 7
nodetext symcov hecid
/*diversion
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 8 )
markersymbol 11
nodes symcov
textcolor 7
nodetext symcov hecid
/*sink
unselect symcov nodes
aselect symcov nodes ( hydronat//hectype = 6 )
markersymbol 23
```

```
nodes symcov
textcolor 7
nodetext symcov hecid
/*
/*
&type
&type HECPREPRO:
&type HECPREPRO: SYMCOV LEGEND:
&type HECPREPRO: ----------------------------------------
&type HECPREPRO:
&type HECPREPRO: LINES:
&type HECPREPRO: White:  Input Stream and Subbasin Lines
&type HECPREPRO: Green:  Reaches
&type HECPREPRO: Yellow: Links
&type HECPREPRO:
&type HECPREPRO: SYMBOLS:
&type HECPREPRO: Squares:              Subbasins
&type HECPREPRO: Stars:                Sources
&type HECPREPRO: Triangles:            Junctions
&type HECPREPRO: Upsidedown Triangles: Reservoirs
&type HECPREPRO: Circles:              Diversions
&type HECPREPRO: Diamonds:             Sinks
&type HECPREPRO:
&type HECPREPRO: TEXT:
&type HECPREPRO: Yellow: HECID of Elements
&type HECPREPRO:
&type HECPREPRO: ----------------------------------------
&type HECPREPRO:
&type
/*
&return
```

APPENDIX B. ARCVIEW AVENUE PROGRAMS.

B.1. Main Program (hecprepro.ave).

```
'
'-----------------
'-----------------
'--- HECPREPRO ---
'-----------------
'-----------------
'
'----------------------------
'--- creation information ---
'----------------------------
'
'Name: hecprepro.ave
'Version: 4.0.av
'Date: 04/20/97
'Author: Ferdi Hellweger
'        Center for Research in Water Resources
'        The University of Texas at Austin
'        ferdi@crwr.utexas.edu
'        www.ce.utexas.edu/stu/ferdi/
'
'--------------------------
'--- purpose/description ---
'--------------------------
'
'This program creates an HMS basin file.
'
'--------------------------
'--- general set up (1) ---
'--------------------------
'
'--- get view (1.1) ---
'
theview = av.getactivedoc
'
'--- get display ---
'
thedisplay = theview.getdisplay
'
'--- get themes (1.2) ---
'
theactivethemes = theview.getactivethemes
if (theactivethemes.count = 0) then
    msgbox.error("No active themes found", "HECPREPRO")
    exit
end
if (theactivethemes.count = 1) then
    msgbox.error("Only one active theme found", "HECPREPRO")
    exit
end
if (theactivethemes.count > 4) then
    msgbox.error("Too many active themes found", "HECPREPRO")
    exit
end
ilfound = false
ipfound = false
infound = false
igfound = false
for each activetheme in theactivethemes
    if (activetheme.getclass.getclassname = "ftheme") then
        theftab = activetheme.getftab
        theshapef = theftab.findfield("shape")
        theshape = theftab.returnvalue(theshapef,0)
        if (theshape.getclass.getclassname = "polyline") then
            iltheme = activetheme
            ilfound = true
        end
        if (theshape.getclass.getclassname = "polygon") then
            iptheme = activetheme
```

```
            ipfound = true
        end
        if (theshape.getclass.getclassname = "point") then
            intheme = activetheme
            infound = true
        end
    end
    if (activetheme.getclass.getclassname = "gtheme") then
        igtheme = activetheme
        igfound = true
    end
end
if (not ilfound) then
    msgbox.error("One theme needs to be a line theme.", "HECPREPRO")
    exit
end
if (not ipfound) then
    msgbox.error("One theme needs to be a polygon theme.", "HECPREPRO")
    exit
end
if ((not infound) and (not igfound) and (theactivethemes.count = 3)) then
    msgbox.error("Three themes are active." + nl + "One theme needs to be a
grid or point/node theme.", "HECPREPRO")
    exit
end
if ((not infound) and (theactivethemes.count = 4)) then
    msgbox.error("Four themes are active." + nl + "One theme needs to be a
point/node theme.", "HECPREPRO")
    exit
end
if ((not igfound) and (theactivethemes.count = 4)) then
    msgbox.error("Four themes are active." + nl + "One theme needs to be a
grid theme.", "HECPREPRO")
    exit
end
'
'--- get run control parameters (1.3) ---
'
labels = list.make
labels = labels.add("Transfer Attributes (y/n)")
labels = labels.add("HMS File Path (default, path)")
labels = labels.add("Tolerance")
labels = labels.add("User Observation Level (0-4)")
'
defaults = list.make
defaults = defaults.add("no")
defaults = defaults.add("default")
defaults = defaults.add("10")
defaults = defaults.add("2")
'
inputs = msgbox.multiinput("Enter run control parameters", "HECPREPRO",
labels, defaults)
if (inputs.count = 0) then
    exit
end
'
if (inputs.get(0).left(1).ucase = "N") then
    attrib = false
else
    attrib = true
end
'
hmsmode = "d"
if (not (inputs.get(1).left(3).ucase = "DEF")) then
    hmsmode = inputs.get(1)
end
'
tol = inputs.get(2).asnumber
'
oblevel = inputs.get(3).asnumber
'
```

```
'--- set up hydrologic element type dictionary (1.4) ---
'
'hedict:
'key = hectype id
'value = hectype name
'
hedict = dictionary.make(7 * 2)
'
hedict.add(5, "subbasin")
hedict.add(4, "reach")
hedict.add(3, "junction")
hedict.add(8, "diversion")
hedict.add(10, "reservoir")
hedict.add(7, "source")
hedict.add(6, "sink")
'
'--- set up error handling ---
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Setting up error handling.  "
    av.showmsg(mainmsg)
    av.setstatus(1)
    av.showstopbutton
end
'
errors = 0
'
'--- set up input themes (1.5) ---
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Setting up input themes.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(2)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'input line theme
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Line theme.")
    keepgoing = av.setstatus(3)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
ilftab = iltheme.getftab
if (ilftab = nil) then
    msgbox.error("Can't open input line theme","HECPREPRO")
    exit
end
'
ilfields = list.make
for each ilfrec in ilftab.getfields
    if (ilfrec.getname = "shape") then
        ilshapef = ilfrec
        continue
    else
        ilfields = ilfields.add(ilfrec)
```

```
    end
end
if (ilshapef = nil) then
    msgbox.error("Can't find 'shape' field in input line theme","HECPREPRO")
    exit
end
'
'input polygon theme
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Polygon theme.")
    keepgoing = av.setstatus(4)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
ipftab = iptheme.getftab
if (ipftab = nil) then
    msgbox.error("Can't open input polygon theme","HECPREPRO")
    exit
end
'
ipfields = list.make
for each ipfrec in ipftab.getfields
    if (ipfrec.getname = "shape") then
        ipshapef = ipfrec
        continue
    else
        ipfields = ipfields.add(ipfrec)
    end
end
if (ipshapef = nil) then
    msgbox.error("Can't find 'shape' field in input polygon
theme","HECPREPRO")
    exit
end
'
'input point/node theme
'
if (infound) then
    if (oblevel >= 1) then
        av.showmsg(mainmsg + "Point/node theme.")
        keepgoing = av.setstatus(3)
        if (not keepgoing) then
            message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
            msgbox.info(message, "HECPREPRO")
            av.clearmsg
            av.clearstatus
            exit
        end
    end
    '
    inftab = intheme.getftab
    if (inftab = nil) then
        msgbox.error("Can't open input point/node theme","HECPREPRO")
        exit
    end
    '
    infields = list.make
    for each infrec in inftab.getfields
        if (infrec.getname = "shape") then
            inshapef = infrec
            continue
        else
```

```
            infields = infields.add(infrec)
        end
    end
    if (inshapef = nil) then
        msgbox.error("Can't find 'shape' field in input point/node
theme","HECPREPRO")
        exit
    end
end
'
'input grid theme
'
'doesn't need set up
'
'
'--- get attribute map info (1.6) ---
'
if (attrib) then
    if (oblevel >= 1) then
        av.showmsg(mainmsg + "Attribute transfer tables.")
        keepgoing = av.setstatus(4)
        if (not keepgoing) then
            message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
            msgbox.info(message, "HECPREPRO")
            av.clearmsg
            av.clearstatus
            exit
        end
    end
    '
    attribpossible = false
    '
    'set up attribute maping dictionary
    '
    'mapdict:
    'key = hydrologic element type
    '0 = map attributes?
    '1 = table name
    '2 = table vtab
    '3 = field name string list
    '
    'field name string list:
    '0 hmsfield
    '1 gisfield
    '2 transfer
    '
    'transfer:
    '0 = no transfer
    '1 = transfer, reach attribute total
    '2 = transfer, reach attribute simple average
    '3 = transfer, reach attribute length weighted average
    '
    mapdict = dictionary.make(7 * 3)
    for each herec in hedict
        mapdict.add(herec.clone, {false, nil, nil, nil})
    end
    '
    'define attribute map table names
    '
    mapdict.get("subbasin").set(1, "hecsub.dbf")
    mapdict.get("reach").set(1, "hecreach.dbf")
    mapdict.get("junction").set(1, "hecjunct.dbf")
    mapdict.get("diversion").set(1, "hecdiv.dbf")
    mapdict.get("reservoir").set(1, "hecres.dbf")
    mapdict.get("source").set(1, "hecsourc.dbf")
    mapdict.get("sink").set(1, "hecsink.dbf")
    '
    'get tables
    '
    theproject = av.getproject
```

```
    thedocs = theproject.getdocs
    thetables = list.make
    for each thedoc in thedocs
        if (thedoc.getclass.getclassname = "table") then
             thetables = thetables.add(thedoc.clone)
        end
    end
    '
    'get map attribute map tables
    '
    for each thetable in thetables
        for each herec in hedict
            if (thetable.getname = mapdict.get(herec).get(1)) then
                mapdict.get(herec).set(0, true)
                mapdict.get(herec).set(2, thetable.getvtab.clone)
            end
        end
    end
    '
    'process attribute map tables
    '
    for each herec in hedict
        if (mapdict.get(herec).get(0)) then
            mapdict.get(herec).set(0, false)
            mapvtb = mapdict.get(herec).get(2)
            hmsf = mapvtb.findfield("hmsfield")
            if (hmsf = nil) then
                msgbox.error("Error in " + herec + " attribute map table",
"HECPREPRO")
                exit
            end
            gisf = mapvtb.findfield("gisfield")
            if (gisf = nil) then
                msgbox.error("Error in " + herec + " attribute map table",
"HECPREPRO")
                exit
            end
            traf = mapvtb.findfield("transfer")
            if (traf = nil) then
                msgbox.error("Error in " + herec + " attribute map table",
"HECPREPRO")
                exit
            end
            templist = list.make
            for each maprec in mapvtb
                tra = mapvtb.returnvalue(traf, maprec)
                if (tra > 0) then
                    hms = mapvtb.returnvalue(hmsf, maprec)
                    if (hms = "") then
                        msgbox.error("Error in " + herec + " attribute map
table", "HECPREPRO")
                        exit
                    end
                    gis = mapvtb.returnvalue(gisf, maprec)
                    if (gis = "") then
                        msgbox.error("Error in " + herec + " attribute map
table", "HECPREPRO")
                        exit
                    end
                    if (herec = "subbasin") then
                        testf = ipftab.findfield(gis)
                        if (testf = nil) then
                            msgbox.error("Error in " + herec + " attribute map
table", "HECPREPRO")
                            exit
                        end
                    end
                    if (herec = "reach") then
                        testf = ilftab.findfield(gis)
                        if (testf = nil) then
```

```
                            msgbox.error("Error in " + herec + " attribute map
table", "HECPREPRO")
                            exit
                        end
                    end
                    if ((not (herec = "reach")) and (not (herec = "subbasin"))
and infound) then
                        testf = inftab.findfield(gis)
                        if (testf = nil) then
                            msgbox.error("Error in " + herec + " attribute map
table", "HECPREPRO")
                            exit
                        end
                    end
                    templist.add({hms.clone, gis.clone, tra.clone})
                    mapdict.get(herec).set(0, true)
                    attribpossible = true
                end
            end
            mapdict.get(herec).set(3, templist.clone)
        end
    end
    if (not attribpossible) then
        msgbox.info("Attribute transfer is not possible." + nl + "Using no
attribute transfer option", "HECPREPRO")
        attrib = false
    end
end
'
'----------------------------------------
'--- create hydro line shape file (2) ---
'----------------------------------------
'
'this code is based on a script taken from esri's home page
'called view.intersectthemes.
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Intersecting input themes.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(5)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'--- set up hydro line theme (2.1) ---
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Setting up hydro line theme.")
    keepgoing = av.setstatus(6)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'set up file
'
hlfilename = av.getproject.makefilename("hydrol", "shp")
hlftab = ftab.makenew(hlfilename, polyline)
'
'add fields to hydro line theme
```

```
'
hlfields = list.make
'
hlhectypef = field.make("hectype", #FIELD_DECIMAL, 16, 4)
hlhecidf = field.make("hecid", #FIELD_DECIMAL, 16, 4)
hllhpf = field.make("lhp", #FIELD_DECIMAL, 16, 4)
hlfphpf = field.make("fphp", #FIELD_DECIMAL, 16, 4)
hltphpf = field.make("tphp", #FIELD_DECIMAL, 16, 4)
'
hlfields = hlfields.add(hlhectypef)
hlfields = hlfields.add(hlhecidf)
hlfields = hlfields.add(hllhpf)
hlfields = hlfields.add(hlfphpf)
hlfields = hlfields.add(hltphpf)
'
hlfields = hlfields + ilfields.deepclone
'
hlftab.addfields(hlfields)
'
hlshapef = hlftab.findfield("shape")
'
'make theme editable
'
hlftab.seteditable(true)
'
'--- user observation ---
'
if (oblevel >= 2) then
    hltheme = ftheme.make(hlftab)
    theview.addtheme(hltheme)
    iltheme.setvisible(false)
    hltheme.setvisible(true)
    theview.draw(thedisplay)
end
'
'--- intersect (2.2) ---
'
'user observation
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Processing each polygon.")
    keepgoing = av.setstatus(7)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
theme1 = iltheme
ftab1 = theme1.getftab
shpfld1 = ftab1.findfield("shape")
therecs1 = ftab1.getselection
theoldsel = ftab1.getselection.clone
if (therecs1.count = 0) then
    therecs1 = ftab1
end
'
theme2 = iptheme
ftab2 = theme2.getftab
shpfld2 = ftab2.findfield("shape")
therecs2 = ftab2.getselection
if (therecs2.count = 0) then
    therecs2 = ftab2
end
'
'loop on each polygon
'
```

```
for each arec2 in therecs2
    '
    'get the polygon shape and select all records within that shape
    '
    thesrcshape = ftab2.returnvalue(shpfld2,arec2)
    if (theview.getprojection.isnull) then
        theme1.selectbyshapes({thesrcshape}, #VTAB_SELTYPE_NEW)
    else
        pshp = thesrcshape.returnprojected(theview.getprojection)
        theme1.selectbyshapes({pshp}, #VTAB_SELTYPE_NEW)
    end
    '
    'for each selected record
    '
    for each selrec in ftab1.getselection
        '
        'get the shape of the record
        '
        selectedshape = ftab1.returnvalue(shpfld1,selrec)
        '
        'if the line is wholely within the polygin (no intersection) then
        '
        if (selectedshape.iscontainedin(thesrcshape)) then
            alineshp = selectedshape
        else
            '
            'split the line using the polygon
            '
            alineshp = selectedshape.lineintersection(thesrcshape)
        end
        '
        'add the new record
        '
        theoutrec = hlftab.addrecord
        '
        'set the shape value
        '
        hlftab.setvalue(hlshapef,theoutrec,alineshp)
        '
        'set the line field values
        '
        for each ilfrec in ilfields
            ilf = ftab1.findfield(ilfrec.getname)
            ilvalue = ftab1.returnvalue(ilf, selrec)
            hlf = hlftab.findfield(ilfrec.getname)
            hlftab.setvalue(hlf, theoutrec, ilvalue)
        end
        '
        if (oblevel >= 3) then
           theview.draw(thedisplay)
        end
    end
end
'
'revert to original selection set
'
ftab1.setselection(theoldsel)
ftab1.updateselection
'
'--- user observation ---
'
if (oblevel >= 2) then
    av.run("HECLEGEND", {hltheme})
    theview.draw(thedisplay)
end
'
'----------------------------------------
'--- create hydro point shape file (3) ---
'----------------------------------------
'
if (oblevel >= 1) then
```

```
    mainmsg = "HECPREPRO: Creating hydro point theme.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(25)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'--- set up hydro point theme (3.1) ---
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Setting up hydro point theme.")
    keepgoing = av.setstatus(26)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'set up file
'
hpfilename = av.getproject.makefilename("hydrop", "shp")
hpftab = ftab.makenew(hpfilename, point)
'
'get shape field
'
hpshapef = hpftab.findfield("shape")
'
'add fields to hydro point theme
'
hpfields = list.make
'
hphecidf = field.make("hecid", #FIELD_DECIMAL, 16, 4)
hphectypef = field.make("hectype", #FIELD_DECIMAL, 16, 4)
hprntf = field.make("rnt", #FIELD_CHAR, 16, 4)
hphntf = field.make("hnt", #FIELD_CHAR, 16, 4)
hprlf = field.make("rl", #FIELD_DECIMAL, 16, 4)
hpruplf = field.make("rupl", #FIELD_DECIMAL, 16, 4)
hprdnlf = field.make("rdnl", #FIELD_DECIMAL, 16, 4)
hprrlf = field.make("rrl", #FIELD_DECIMAL, 16, 4)
hprruplf = field.make("rrupl", #FIELD_DECIMAL, 16, 4)
hprrdnlf = field.make("rrdnl", #FIELD_DECIMAL, 16, 4)
hphlf = field.make("hl", #FIELD_DECIMAL, 16, 4)
hphuplf = field.make("hupl", #FIELD_DECIMAL, 16, 4)
hphdnlf = field.make("hdnl", #FIELD_DECIMAL, 16, 4)
hphrlf = field.make("hrl", #FIELD_DECIMAL, 16, 4)
hphruplf = field.make("hrupl", #FIELD_DECIMAL, 16, 4)
hphrdnlf = field.make("hrdnl", #FIELD_DECIMAL, 16, 4)
'
hpfields = hpfields.add(hphecidf)
hpfields = hpfields.add(hphectypef)
hpfields = hpfields.add(hprntf)
hpfields = hpfields.add(hphntf)
hpfields = hpfields.add(hprlf)
hpfields = hpfields.add(hpruplf)
hpfields = hpfields.add(hprdnlf)
hpfields = hpfields.add(hprrlf)
hpfields = hpfields.add(hprruplf)
hpfields = hpfields.add(hprrdnlf)
hpfields = hpfields.add(hphlf)
hpfields = hpfields.add(hphuplf)
hpfields = hpfields.add(hphdnlf)
```

```
hpfields = hpfields.add(hphrlf)
hpfields = hpfields.add(hphruplf)
hpfields = hpfields.add(hphrdnlf)
'
if (infound) then
    hpfields = hpfields + infields.deepclone
end
'
hpftab.addfields(hpfields)
'
'make hydro point theme editable
'
hpftab.seteditable(true)
'
'--- user observation ---
'
if (oblevel >= 2) then
    hptheme = ftheme.make(hpftab)
    theview.addtheme(hptheme)
    hptheme.setvisible(true)
    theview.draw(thedisplay)
end
'
'--- add points (3.2) ---
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Adding points.")
    keepgoing = av.setstatus(27)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'loop for each line
'
for each hlrec in hlftab
    '
    hlshape = hlftab.returnvalue(hlshapef, hlrec)
    hlpoints = hlshape.asmultipoint.aslist
    if (hlpoints.count > 1) then
        hlfp = hlpoints.get(0)
        hltp = hlpoints.get(hlpoints.count - 1)
        '
        'add fpoint
        '
        exists = false
        for each hprec in hpftab
            hpshape = hpftab.returnvalue(hpshapef, hprec)
            if (hpshape.distance(hlfp) < tol) then
                exists = true
                hlfphp = hprec.clone
                break
            end
        end
        if (not exists) then
            theoutrec = hpftab.addrecord
            hpftab.setvalue(hpshapef, theoutrec, hlfp)
            hlftab.setvalue(hlfphpf, hlrec, theoutrec)
            if (oblevel >= 3) then
                theview.draw(thedisplay)
            end
            if (infound) then
                for each inrec in inftab
                    inshape = inftab.returnvalue(inshapef, inrec)
                    if (inshape.distance(hlfp) < tol) then
                        for each infrec in inftab.getfields
```

```
                            if (infrec.getname = "shape") then
                                continue
                            else
                                inval = inftab.returnvalue(infrec, inrec)
                                hpf = hpftab.findfield(infrec.getname)
                                hpftab.setvalue(hpf, theoutrec, inval)
                            end
                        end
                        break
                    end
                end
            end
        end
        if (exists) then
            hlftab.setvalue(hlfphpf, hlrec, hlfphp)
        end
        '
        'add tpoint
        '
        exists = false
        for each hprec in hpftab
            hpshape = hpftab.returnvalue(hpshapef, hprec)
            if (hpshape.distance(hltp) < tol) then
                exists = true
                hltphp = hprec.clone
                break
            end
        end
        if (not exists) then
            theoutrec = hpftab.addrecord
            hpftab.setvalue(hpshapef, theoutrec, hltp)
            hlftab.setvalue(hltphpf, hlrec, theoutrec)
            if (oblevel >= 3) then
                theview.draw(thedisplay)
            end
            if (infound) then
                for each inrec in inftab
                    inshape = inftab.returnvalue(inshapef, inrec)
                    if (inshape.distance(hltp) < tol) then
                        for each infrec in inftab.getfields
                            if (infrec.getname = "shape") then
                                continue
                            else
                                inval = inftab.returnvalue(infrec, inrec)
                                hpf = hpftab.findfield(infrec.getname)
                                hpftab.setvalue(hpf, theoutrec, inval)
                            end
                        end
                        break
                    end
                end
            end
        end
        if (exists) then
            hlftab.setvalue(hltphpf, hlrec, hltphp)
        end
    end
end
'
'--- user observation ---
'
if (oblevel >= 2) then
    av.run("HECLEGEND", {hptheme})
    theview.draw(thedisplay)
end
'
'-------------------------------------------------------
'--- identify sources, subbasin outlets and sinks (4) ---
'-------------------------------------------------------
'
'--- user observation ---
```

```
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Identifying sources, subbasin outlets and sinks.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(50)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'--- identify and mark points ---
'
for each hprec in hpftab
    hpshape = hpftab.returnvalue(hpshapef, hprec)
    for each polyrec in ipftab
        polyshape = ipftab.returnvalue(ipshapef, polyrec)
        polylshape = polyshape.aspolyline
        if (hpshape.distance(polylshape) < tol) then
            hpftab.setvalue(hphectypef, hprec, 1)
            if (oblevel >= 3) then
                av.run("HECLEGEND", {hptheme})
                theview.draw(thedisplay)
            end
        end
    end
end
'
'--- user observation ---
'
if (oblevel >= 2) then
    av.run("HECLEGEND", {hptheme})
    theview.draw(thedisplay)
end
'
'-----------------------------------
'--- identify channel system (5) ---
'-----------------------------------
'
'--- user observation ---
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Identifying channel system.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(60)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'--- create stream dictionary (5.1) ---
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Creating stream dictionary.")
    keepgoing = av.setstatus(61)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
```

```
    end
end
'
'streamdict:
'key = hlrec
'0 = hlrec
'
streamdict = dictionary.make(hlftab.getnumrecords * 2)
'
for each hlrec in hlftab
    if (oblevel >= 1) then
        av.showmsg(mainmsg + "Line number " + hlrec.asstring)
    end
    hlshape = hlftab.returnvalue(hlshapef, hlrec)
    hlpoints = hlshape.asmultipoint.aslist
    if (hlpoints.count < 2) then
        hllength = 0
    else
        hlfp = hlpoints.get(0)
        hltp = hlpoints.get(hlpoints.count - 1)
        hllength = (((hltp.getx - hlfp.getx)^2) + ((hltp.gety -
hlfp.gety)^2))^0.5
    end
    if (hllength > (tol / 2)) then
        streamdict.add(hlrec.clone, hlrec.clone)
    end
end
'
'--- create channel dictionary (5.2) ---
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Creating channel dictionary.")
    keepgoing = av.setstatus(62)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'channeldict:
'key = hlrec
'0 = hlrec
'
channeldict = dictionary.make(hlftab.getnumrecords * 2)
'
uplist = list.make
for each hprec in hpftab
    if (hpftab.returnvalue(hphectypef, hprec) = 1) then
        hpshape = hpftab.returnvalue(hpshapef, hprec)
        for each hlrec in streamdict
            hlshape = hlftab.returnvalue(hlshapef, hlrec)
            hlpoints = hlshape.asmultipoint.aslist
            if (hlpoints.get(0).distance(hpshape) < tol) then
                channeldict.add(hlrec.clone, hlrec.clone)
                uplist = uplist.add(hlrec.clone)
            end
        end
        for each hlrec in channeldict
            streamdict.remove(hlrec)
        end
    end
end
'
done = false
while (not done)
    dnlist = list.make
    for each uplrec in uplist
```

```
        uplineshape = hlftab.returnvalue(hlshapef, uplrec)
        uplinepoints = uplineshape.asmultipoint.aslist
        updnpoint = uplinepoints.get(uplinepoints.count - 1)
        for each hlrec in streamdict
           hlshape = hlftab.returnvalue(hlshapef, hlrec)
           if (hlshape.asmultipoint.aslist.get(0).distance(updnpoint) < tol)
then
               dnlist = dnlist.add(hlrec.clone)
               channeldict.add(hlrec.clone, hlrec.clone)
           end
        end
        for each hlrec in channeldict
            streamdict.remove(hlrec)
        end
    end
    if (dnlist.count = 0) then
        done = true
    end
    uplist = dnlist
end
'
'--- mark lines (5.3) ---
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Marking lines.")
    keepgoing = av.setstatus(69)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
for each hlrec in channeldict
    hlftab.setvalue(hlhectypef, hlrec, 4)
    if (oblevel >= 3) then
        av.run("HECLEGEND", {hltheme})
        theview.draw(thedisplay)
    end
end
'
'--- user observation ---
'
if (oblevel >= 2) then
    av.run("HECLEGEND", {hltheme})
    theview.draw(thedisplay)
end
'
'--------------------------
'--- identify lakes (6) ---
'--------------------------
'
'--- user observation ---
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Identifying lakes.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(70)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
```

```
'--- identify lines ---
'
for each hprec in hpftab
    uplines = 0
    dnlines = 0
    dnlinelist = list.make
    for each hlrec in hlftab
        hlshape = hlftab.returnvalue(hlshapef, hlrec)
        hlpointlist = hlshape.asmultipoint.aslist
        if (hlpointlist.count < 2) then
            hllength = 0
        else
            fpoint = hlpointlist.get(0)
            tpoint = hlpointlist.get(hlpointlist.count - 1)
            hllength = (((tpoint.getx - fpoint.getx)^2) + ((tpoint.gety -
fpoint.gety)^2))^0.5
        end
        if (hllength > (tol / 2)) then
            hlfphp = hlftab.returnvalue(hlfphpf, hlrec)
            hltphp = hlftab.returnvalue(hltphpf, hlrec)
            if (hlfphp = hprec) then
                dnlines = dnlines + 1
                dnlinelist = dnlinelist.add(hlrec.clone)
            end
            if (hltphp = hprec) then
                uplines = uplines + 1
            end
        end
    end
    if ((uplines = 1) and (dnlines = 2)) then
        tracelist = list.make
        tracelist.add(dnlinelist.get(0).clone)
        tracelist.add(dnlinelist.get(1).clone)
        uplist = dnlinelist.clone
        done = false
        while (not done)
            dnlist = list.make
            for each uplrec in uplist
                upltphp = hlftab.returnvalue(hltphpf, uplrec)
                for each hlrec in hlftab
                    hlfphp = hlftab.returnvalue(hlfphpf, hlrec)
                    if (hlfphp = upltphp) then
                        dnlist = dnlist.add(hlrec.clone)
                        tracelist.add(hlrec.clone)
                    end
                end
            end
            if (dnlist.count = 0) then
                done = true
            end
            uplist = dnlist
        end
        tracelistb = tracelist.clone
        tracelistb.removeduplicates
        lake = false
        i = 0
        for each hlrec in tracelist
            if (not (tracelist.get(i) = tracelistb.get(i))) then
                lake = true
                dnline = tracelist.get(i)
                break
            end
            i = i + 1
        end
        if (lake) then
            dnphp = hlftab.returnvalue(hlfphpf, dnline)
            tracelistc = list.make
            for each hlrec in tracelist
                hltphp = hlftab.returnvalue(hltphpf, hlrec)
                if (hltphp = dnphp) then
                    dnline = hlrec.clone
```

```
                    tracelistc.add(hlrec.clone)
                    done = false
                    while (not done)
                        dnfphp =  hlftab.returnvalue(hlfphpf, dnline)
                        found = false
                        for each hlrec2 in tracelist
                           hl2tphp = hlftab.returnvalue(hltphpf, hlrec2)
                           if (hl2tphp = dnfphp) then
                               found = true
                               upline = hlrec2.clone
                               tracelistc.add(hlrec2.clone)
                           end
                        end
                        if (found = false) then
                            done = true
                        end
                        dnline = upline
                    end
                end
            end
            for each hlrec in tracelistc
                hlhectype = hlftab.returnvalue(hlhectypef, hlrec)
                hlhectype = hlhectype * 10
                hlftab.setvalue(hlhectypef, hlrec, hlhectype)
                hlftab.setvalue(hllhpf, hlrec, dnphp)
                if (oblevel >= 3) then
                    av.run("HECLEGEND", {hltheme})
                    theview.draw(thedisplay)
                end
            end
        end
    end
end
'
'--- user observation ---
'
if (oblevel >= 2) then
    av.run("HECLEGEND", {hltheme})
    theview.draw(thedisplay)
end
'
'-------------------------------------
'--- identify channel elements (7) ---
'-------------------------------------
'
'--- user observation ---
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Identifying channel elements.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(80)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'--- set up sym dictionaries (7.1) ---
'
'sldict:
'key = hecid
'0 = hecid
'1 = hectype
'2 = hecupid
'3 = hecdnid
'4 = hecupx
'5 = hecupy
```

```
'6 = hecdnx
'7 = hecdny
'
'spdict:
'key = hecid
'0 = hecid
'1 = hectype
'2 = hecupids
'3 = hecdnids
'4 = hecx
'5 = hecy
'6 = hprec
'
'upiddict:
'key = hecid
'0 = hecid
'1 = hecupid 1
'2 = hecupid 2
'3 = etc...
'
'dniddict:
'key = hecid
'0 = hecid
'1 = hecdnid 1
'2 = hecdnid 2
'3 = etc...
'
sldict = dictionary.make(hlftab.getnumrecords * 8)
spdict = dictionary.make(hpftab.getnumrecords * 7)
upiddict = dictionary.make(hpftab.getnumrecords * 2)
dniddict = dictionary.make(hpftab.getnumrecords * 2)
'
'--- set up hecid variable (7.2) ---
'
hecid = 1
'
'--- identify elements (7.3) ---
'
for each hprec in hpftab
    if (oblevel >= 1) then
        av.showmsg(mainmsg + "Point number " + hprec.asstring)
    end
    hpshape = hpftab.returnvalue(hpshapef, hprec)
    hphecx = hpshape.getx
    hphecy = hpshape.gety
    hphectype = hpftab.returnvalue(hphectypef, hprec)
    rlines = 0
    ruplines = 0
    rdnlines = 0
    rrlines = 0
    rruplines = 0
    rrdnlines = 0
    hlines = 0
    huplines = 0
    hdnlines = 0
    hrlines = 0
    hruplines = 0
    hrdnlines = 0
    for each hlrec in hlftab
        hlhectype = hlftab.returnvalue(hlhectypef, hlrec)
        hlshape = hlftab.returnvalue(hlshapef, hlrec)
        hlpoints = hlshape.asmultipoint.aslist
        if (hlpoints.count < 2) then
            hllength = 0
        else
            hlfp = hlpoints.get(0)
            hltp = hlpoints.get(hlpoints.count - 1)
            hllength = (((hltp.getx - hlfp.getx)^2) + ((hltp.gety -
hlfp.gety)^2))^0.5
        end
        if (hllength > (tol / 2)) then
```

```
        if (hlfp.distance(hpshape) < tol) then
            rlines = rlines + 1
            rdnlines = rdnlines + 1
            if (hlhectype > 10 ) then
                rrlines = rrlines + 1
                rrdnlines = rrdnlines + 1
            end
            if ((hlhectype = 4) or (hlhectype = 40)) then
                hlines = hlines + 1
                hdnlines = hdnlines + 1
                if (hlhectype = 40 ) then
                    hrlines = hrlines + 1
                    hrdnlines = hrdnlines + 1
                end
            end
        else
            if (hltp.distance(hpshape) < tol) then
                rlines = rlines + 1
                ruplines = ruplines + 1
                if (hlhectype > 10 ) then
                    rrlines = rrlines + 1
                    rruplines = rruplines + 1
                end
                if ((hlhectype = 4) or (hlhectype = 40)) then
                    hlines = hlines + 1
                    huplines = huplines + 1
                    if (hlhectype = 40 ) then
                        hrlines = hrlines + 1
                        hruplines = hruplines + 1
                    end
                end
            end
        end
    end
end
'
'calculate node types
'
rnodetype = "unknown"
if ((rlines = 1) and (rdnlines = 1)) then
    rnodetype = "updangling"
end
if ((rlines = 1) and (ruplines = 1)) then
    rnodetype = "dndangling"
end
if ((ruplines = 1) and (rdnlines = 1)) then
    rnodetype = "interior"
end
if ((ruplines >= 2) and (rdnlines = 1)) then
    rnodetype = "junction"
end
if ((ruplines = 1) and (rdnlines >= 2)) then
    rnodetype = "diversion"
end
if ((rruplines = 1) and (rrdnlines = 1)) then
    rnodetype = "mreservoir"
end
if ((rrdnlines = 2) and (rruplines = 0)) then
    rnodetype = "upreservoir"
end
if ((rruplines = 2) and (rrdnlines = 0)) then
    rnodetype = "dnreservoir"
end
if ((rruplines = 1) and (rrdnlines = 1) and (rdnlines = 2)) then
    rnodetype = "dreservoir"
end
if ((rruplines = 1) and (rrdnlines = 1) and (ruplines = 2)) then
    rnodetype = "jreservoir"
end
'
hnodetype = "unknown"
```

```
if ((hlines = 1) and (hdnlines = 1)) then
    hnodetype = "updangling"
end
if ((hlines = 1) and (huplines = 1)) then
    hnodetype = "dndangling"
end
if ((huplines = 1) and (hdnlines = 1)) then
    hnodetype = "interior"
end
if ((huplines >= 2) and (hdnlines = 1)) then
    hnodetype = "junction"
end
if ((huplines = 1) and (hdnlines >= 2)) then
    hnodetype = "diversion"
end
if ((hruplines = 1) and (hrdnlines = 1)) then
    hnodetype = "mreservoir"
end
if ((hrdnlines = 2) and (hruplines = 0)) then
    hnodetype = "upreservoir"
end
if ((hruplines = 2) and (hrdnlines = 0)) then
    hnodetype = "dnreservoir"
end
if ((hruplines = 1) and (hrdnlines = 1) and (hdnlines = 2)) then
    hnodetype = "dreservoir"
end
if ((hruplines = 1) and (hrdnlines = 1) and (huplines = 2)) then
    hnodetype = "jreservoir"
end
if (hlines = 0) then
    hnodetype = "none"
end
if (oblevel >= 4) then
    message = "rlines = " + rlines.asstring
    message = message + nl + "ruplines = " + ruplines.asstring
    message = message + nl + "rdnlines = " + rdnlines.asstring
    message = message + nl + "rnodetype = " + rnodetype
    message = message + nl + "hlines = " + hlines.asstring
    message = message + nl + "huplines = " + huplines.asstring
    message = message + nl + "hdnlines = " + hdnlines.asstring
    message = message + nl + "hnodetype = " + hnodetype
    msgbox.report(message, "HECPREPRO")
end
if (rnodetype = "unknown") then
    if (oblevel >= 2) then
        msgbox.error("Unidentifiable rnodetype", "HECPREPRO")
    end
    errors = errors + 1
end
if (hnodetype = "unknown") then
    if (oblevel >= 2) then
        msgbox.error("Unidentifiable hnodetype", "HECPREPRO")
    end
    errors = errors + 1
end
'
'record node types
'
hpftab.setvalue(hprntf, hprec, rnodetype)
hpftab.setvalue(hphntf, hprec, hnodetype)
hpftab.setvalue(hprlf, hprec, rlines)
hpftab.setvalue(hpruplf, hprec, ruplines)
hpftab.setvalue(hprdnlf, hprec, rdnlines)
hpftab.setvalue(hprrlf, hprec, rrlines)
hpftab.setvalue(hprruplf, hprec, rruplines)
hpftab.setvalue(hprrdnlf, hprec, rrdnlines)
hpftab.setvalue(hphlf, hprec, hlines)
hpftab.setvalue(hphuplf, hprec, huplines)
hpftab.setvalue(hphdnlf, hprec, hdnlines)
hpftab.setvalue(hphrlf, hprec, hrlines)
```

```
    hpftab.setvalue(hphruplf, hprec, hruplines)
    hpftab.setvalue(hphrdnlf, hprec, hrdnlines)
    '
    'classify elements based on hectype system
    '
    if ((hnodetype = "junction") and (not (hphectype = 1))) then
        hphectype = 3
        '
        hpftab.setvalue(hphectypef, hprec, hphectype)
        hpftab.setvalue(hphecidf, hprec, hecid)
        '
        spdictrec = {hecid.clone, hphectype.clone, 0, 0, hpshape.getx.clone,
hpshape.gety.clone, hprec.clone}
        spdict.add(hecid.clone, spdictrec.clone)
        upiddict.add(hecid.clone, {})
        dniddict.add(hecid.clone, {})
        '
        if (oblevel >= 3) then
            av.run("HECLEGEND", {hptheme})
            theview.draw(thedisplay)
        end
        '
        if (oblevel >= 4) then
            message = "Recording HECID, HECTYPE, HECX and HECY"
            message = message + nl
            message = message + nl + "JUNCTION"
            message = message + nl + "HECID: " + hecid.asstring
            message = message + nl + "HECTYPE: " + hphectype.asstring
            message = message + nl + "HECX: " + hphecx.asstring
            message = message + nl + "HECY: " + hphecy.asstring
            msgbox.report(message, "HECPREPRO")
        end
        '
        hecid = hecid + 1
    end
    if (hnodetype = "diversion") then
        hphectype = 8
        '
        hpftab.setvalue(hphectypef, hprec, hphectype)
        hpftab.setvalue(hphecidf, hprec, hecid)
        '
        spdictrec = {hecid.clone, hphectype.clone, 0, 0, hpshape.getx.clone,
hpshape.gety.clone, hprec.clone}
        spdict.add(hecid.clone, spdictrec.clone)
        upiddict.add(hecid.clone, {})
        dniddict.add(hecid.clone, {})
        '
        if (oblevel >= 3) then
            av.run("HECLEGEND", {hptheme})
            theview.draw(thedisplay)
        end
        '
        if (oblevel >= 4) then
            message = "Recording HECID, HECTYPE, HECX and HECY"
            message = message + nl
            message = message + nl + "DIVERSION"
            message = message + nl + "HECID: " + hecid.asstring
            message = message + nl + "HECTYPE: " + hphectype.asstring
            message = message + nl + "HECX: " + hphecx.asstring
            message = message + nl + "HECY: " + hphecy.asstring
            msgbox.report(message, "HECPREPRO")
        end
        '
        hecid = hecid + 1
    end
    if (rnodetype = "dndangling") then
        hphectype = 6
        '
        hpftab.setvalue(hphectypef, hprec, hphectype)
        hpftab.setvalue(hphecidf, hprec, hecid)
        '
```

```
        spdictrec = {hecid.clone, hphectype.clone, 0, 0, hpshape.getx.clone,
hpshape.gety.clone, hprec.clone}
        spdict.add(hecid.clone, spdictrec.clone)
        upiddict.add(hecid.clone, {})
        dniddict.add(hecid.clone, {})
        '
        if (oblevel >= 3) then
            av.run("HECLEGEND", {hptheme})
            theview.draw(thedisplay)
        end
        '
        if (oblevel >= 4) then
            message = "Recording HECID, HECTYPE, HECX and HECY"
            message = message + nl
            message = message + nl + "SINK"
            message = message + nl + "HECID: " + hecid.asstring
            message = message + nl + "HECTYPE: " + hphectype.asstring
            message = message + nl + "HECX: " + hphecx.asstring
            message = message + nl + "HECY: " + hphecy.asstring
            msgbox.report(message, "HECPREPRO")
        end
        '
        hecid = hecid + 1
    end
    if ((rnodetype = "updangling") and (hnodetype = "updangling")) then
        hphectype = 7
        '
        hpftab.setvalue(hphectypef, hprec, hphectype)
        hpftab.setvalue(hphecidf, hprec, hecid)
        '
        spdictrec = {hecid.clone, hphectype.clone, 0, 0, hpshape.getx.clone,
hpshape.gety.clone, hprec.clone}
        spdict.add(hecid.clone, spdictrec.clone)
        upiddict.add(hecid.clone, {})
        dniddict.add(hecid.clone, {})
        '
        if (oblevel >= 3) then
            av.run("HECLEGEND", {hptheme})
            theview.draw(thedisplay)
        end
        '
        if (oblevel >= 4) then
            message = "Recording HECID, HECTYPE, HECX and HECY"
            message = message + nl
            message = message + nl + "SOURCE"
            message = message + nl + "HECID: " +hecid.asstring
            message = message + nl + "HECTYPE: " + hphectype.asstring
            message = message + nl + "HECX: " + hphecx.asstring
            message = message + nl + "HECY: " + hphecy.asstring
            msgbox.report(message, "HECPREPRO")
        end
        '
        hecid = hecid + 1
    end
    if (hnodetype = "dnreservoir") then
        hphectype = 10
        '
        hpftab.setvalue(hphectypef, hprec, hphectype)
        hpftab.setvalue(hphecidf, hprec, hecid)
        '
        spdictrec = {hecid.clone, hphectype.clone, 0, 0, hpshape.getx.clone,
hpshape.gety.clone, hprec.clone}
        spdict.add(hecid.clone, spdictrec.clone)
        upiddict.add(hecid.clone, {})
        dniddict.add(hecid.clone, {})
        '
        if (oblevel >= 3) then
            av.run("HECLEGEND", {hptheme})
            theview.draw(thedisplay)
        end
        '
```

```
        if (oblevel >= 4) then
            message = "Recording HECID, HECTYPE, HECX and HECY"
            message = message + nl
            message = message + nl + "RESERVOIR"
            message = message + nl + "HECID: " + hecid.asstring
            message = message + nl + "HECTYPE:" + hphectype.asstring
            message = message + nl + "HECX: " + hphecx.asstring
            message = message + nl + "HECY: " + hphecy.asstring
            msgbox.report(message, "HECPREPRO")
        end
        '
        hecid = hecid + 1
    end
    if (hphectype = 1) then
        hpftab.setvalue(hphecidf, hprec, hecid)
        '
        spdictrec = {hecid.clone, hphectype.clone, 0, 0, hpshape.getx.clone,
hpshape.gety.clone, hprec.clone}
        spdict.add(hecid.clone, spdictrec.clone)
        upiddict.add(hecid.clone, {})
        dniddict.add(hecid.clone, {})
        '
        if (oblevel >= 3) then
            av.run("HECLEGEND", {hptheme})
            theview.draw(thedisplay)
        end
        '
        if (oblevel >= 4) then
            message = "Recording HECID, HECTYPE, HECX and HECY"
            message = message + nl
            message = message + nl + "SUBBASIN OUTLET"
            message = message + nl + "HECID: " + hecid.asstring
            message = message + nl + "HECTYPE: " + hphectype.asstring
            message = message + nl + "HECX: " + hphecx.asstring
            message = message + nl + "HECY: " + hphecy.asstring
            msgbox.report(message, "HECPREPRO")
        end
        '
        hecid = hecid + 1
    end
end
'
'--- user observation ---
'
if (oblevel >= 2) then
    av.run("HECLEGEND", {hptheme})
    theview.draw(thedisplay)
end
'
'---------------------------------
'--- establish connectivity (8) ---
'---------------------------------
'
'--- user observation ---
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Establishing connectivity.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(90)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
'--- reaches and channel elements (8.1) ---
'
```

```
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Reaches and channel elements.")
    keepgoing = av.setstatus(91)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
processdict = channeldict.clone
startdict = channeldict.clone
for each hlrec in processdict
    hllhp = hlftab.returnvalue(hllhpf, hlrec)
    if (hllhp > 0) then
        startdict.remove(hlrec)
    end
end
'
eof = false
while (not eof)
    reachlist = list.make
    reachlist.add(startdict.aslist.get(0).clone)
    '
    'go upstream
    '
    if (oblevel >= 1) then
        av.showmsg(mainmsg + "Looking for upstream end of reach.")
    end
    '
    dnline = reachlist.get(0).clone
    '
    upfound = false
    uplake = false
    '
    hlfphp = hlftab.returnvalue(hlfphpf, dnline)
    uphectype = hpftab.returnvalue(hphectypef, hlfphp)
    '
    if (uphectype = 1) then
        upfound = true
    end
    if (uphectype = 3) then
        upfound = true
    end
    if (uphectype = 7) then
        upfound = true
    end
    if (uphectype = 8) then
        upfound = true
    end
    if (uphectype = 10) then
        upfound = true
    end
    if (oblevel >= 4) then
        message = "Looking for upstream end of reach."
        message = message + nl
        message = message + nl + "HLFPHP: " + hlfphp.asstring
        message = message + nl + "UPHECTYPE: " + uphectype.asstring
        message = message + nl + "UPFOUND: " + upfound.asstring
        message = message + nl + "UPLAKE: " + uplake.asstring
        message = message + nl + "REACHLIST: " + reachlist.count.asstring
        message = message + nl + "PROCESSDICT: " + processdict.count.asstring
        message = message + nl + "CHANNELDICT: " + channeldict.count.asstring
        msgbox.report(message, "HECPREPRO")
    end
    while ((not upfound) and (not uplake))
        for each prec in processdict
            hltphp = hlftab.returnvalue(hltphpf, prec)
```

```
            if (hltphp = hlfphp) then
                hllhp = hlftab.returnvalue(hllhpf, prec)
                if (hllhp = 0) then
                    reachlist.add(prec.clone)
                end
                if (hllhp > 0) then
                    uplake = true
                end
                dnline = prec.clone
                break
            end
        end
        hlfphp = hlftab.returnvalue(hlfphpf, dnline)
        uphectype = hpftab.returnvalue(hphectypef, hlfphp)
        if (uphectype = 1) then
            upfound = true
        end
        if (uphectype = 3) then
            upfound = true
        end
        if (uphectype = 7) then
            upfound = true
        end
        if (uphectype = 8) then
            upfound = true
        end
        if (uphectype = 10) then
            upfound = true
        end
        if (oblevel >= 4) then
            message = "Looking for upstream end of reach."
            message = message + nl
            message = message + nl + "HLFPHP: " + hlfphp.asstring
            message = message + nl + "UPHECTYPE: " + uphectype.asstring
            message = message + nl + "UPFOUND: " + upfound.asstring
            message = message + nl + "UPLAKE: " + uplake.asstring
            message = message + nl + "REACHLIST: " + reachlist.count.asstring
            message = message + nl + "PROCESSDICT: " +
processdict.count.asstring
            message = message + nl + "CHANNELDICT: " +
channeldict.count.asstring
            msgbox.report(message, "HECPREPRO")
        end
    end
    if (not uplake) then
        hecupid = hpftab.returnvalue(hphecidf, hlfphp)
        hecupshape = hpftab.returnvalue(hpshapef, hlfphp)
        hecupx = hecupshape.getx
        hecupy = hecupshape.gety
    end
    if (uplake) then
        hecupid = hpftab.returnvalue(hphecidf, hllhp)
        hecupshape = hpftab.returnvalue(hpshapef, hllhp)
        hecupx = hecupshape.getx
        hecupy = hecupshape.gety
    end
    for each hlrec in reachlist
        processdict.remove(hlrec)
        startdict.remove(hlrec)
    end
    '
    'go dnstream
    '
    upline = reachlist.get(0).clone
    '
    dnfound = false
    dnlake = false
    '
    hltphp = hlftab.returnvalue(hltphpf, upline)
    dnhectype = hpftab.returnvalue(hphectypef, hltphp)
    '
```

```
if (dnhectype = 1) then
    dnfound = true
end
if (dnhectype = 3) then
    dnfound = true
end
if (dnhectype = 6) then
    dnfound = true
end
if (dnhectype = 8) then
    dnfound = true
end
if (dnhectype = 10) then
    dnfound = true
end
if (oblevel >= 4) then
    message = "Looking for downstream end of reach."
    message = message + nl
    message = message + nl + "HLTPHP: " + hltphp.asstring
    message = message + nl + "DNHECTYPE: " + dnhectype.asstring
    message = message + nl + "DNFOUND: " + dnfound.asstring
    message = message + nl + "DNLAKE: " + dnlake.asstring
    message = message + nl + "REACHLIST: " + reachlist.count.asstring
    message = message + nl + "PROCESSDICT: " + processdict.count.asstring
    message = message + nl + "CHANNELDICT: " + channeldict.count.asstring
    msgbox.report(message, "HECPREPRO")
end
while ((not dnfound) and (not dnlake))
    if (not dnfound) then
        for each prec in processdict
            hlfphp = hlftab.returnvalue(hlfphpf, prec)
            if (hlfphp = hltphp) then
                hllhp = hlftab.returnvalue(hllhpf, prec)
                if (hllhp = 0) then
                    reachlist.add(prec.clone)
                end
                if (hllhp > 0) then
                    dnlake = true
                end
                upline = prec.clone
                break
            end
        end
    end
    hltphp = hlftab.returnvalue(hltphpf, upline)
    dnhectype = hpftab.returnvalue(hphectypef, hltphp)
    '
    if (dnhectype = 1) then
        dnfound = true
    end
    if (dnhectype = 3) then
        dnfound = true
    end
    if (dnhectype = 6) then
        dnfound = true
    end
    if (dnhectype = 8) then
        dnfound = true
    end
    if (dnhectype = 10) then
        dnfound = true
    end
    if (oblevel >= 4) then
        message = "Looking for downstream end of reach."
        message = message + nl
        message = message + nl + "HLTPHP: " + hltphp.asstring
        message = message + nl + "DNHECTYPE: " + dnhectype.asstring
        message = message + nl + "DNFOUND: " + dnfound.asstring
        message = message + nl + "DNLAKE: " + dnlake.asstring
        message = message + nl + "REACHLIST: " + reachlist.count.asstring
```

```
            message = message + nl + "PROCESSDICT: " +
processdict.count.asstring
            message = message + nl + "CHANNELDICT: " +
channeldict.count.asstring
            msgbox.report(message, "HECPREPRO")
        end
    end
    if (not dnlake) then
        hecdnid = hpftab.returnvalue(hphecidf, hltphp)
        hecdnshape = hpftab.returnvalue(hpshapef, hltphp)
        hecdnx = hecdnshape.getx
        hecdny = hecdnshape.gety
    end
    if (dnlake) then
        hecdnid = hpftab.returnvalue(hphecidf, hllhp)
        hecdnshape = hpftab.returnvalue(hpshapef, hllhp)
        hecdnx = hecdnshape.getx
        hecdny = hecdnshape.gety
    end
    for each hlrec in reachlist
        processdict.remove(hlrec)
        startdict.remove(hlrec)
    end
    if (oblevel >= 4) then
        message = "Recording reach data."
        message = message + nl
        message = message + nl + "HECID: " + hecid.asstring
        message = message + nl + "HECUPID: " + hecupid.asstring
        message = message + nl + "HECUPX: " + hecupx.asstring
        message = message + nl + "HECUPY: " + hecupy.asstring
        message = message + nl + "HECDNID: " + hecdnid.asstring
        message = message + nl + "HECDNX: " + hecdnx.asstring
        message = message + nl + "HECDNY: " + hecdny.asstring
        message = message + nl + "REACHLIST: " + reachlist.count.asstring
        msgbox.report(message, "HECPREPRO")
    end
    '
    'write hydro line theme
    '
    for each hlrec in reachlist
        hlftab.setvalue(hlhecidf, hlrec, hecid.clone)
    end
    '
    'write data to sym line dictionary
    '
    sldictrec = list.make
    sldictrec = sldictrec.add(hecid.clone)
    sldictrec = sldictrec.add(4)
    sldictrec = sldictrec.add(hecupid.clone)
    sldictrec = sldictrec.add(hecdnid.clone)
    sldictrec = sldictrec.add(hecupx.clone)
    sldictrec = sldictrec.add(hecupy.clone)
    sldictrec = sldictrec.add(hecdnx.clone)
    sldictrec = sldictrec.add(hecdny.clone)
    sldict.add(hecid.clone, sldictrec.clone)
    '
    'write data to sym point dictionary
    '
    'upstream element
    '
    sphecdnids = spdict.get(hecupid).get(3) + 1
    spdict.get(hecupid).set(3, sphecdnids)
    sphecdnid = hecid.clone
    dniddict.get(hecupid).add(sphecdnid.clone)
    '
    'downstream element
    '
    sphecupids = spdict.get(hecupid).get(2) + 1
    spdict.get(hecdnid).set(2, sphecupids)
    sphecupid = hecid.clone
    upiddict.get(hecdnid).add(sphecupid.clone)
```

```
    '
    'keep hecid counter going
    '
    hecid = hecid + 1
    '
    'close loop
    '
    if (startdict.count = 0) then
        eof = true
    end
end
'
'--- subbasins (8.2) ---
'
if (oblevel >= 1) then
    av.showmsg(mainmsg + "Subbasins.")
    keepgoing = av.setstatus(95)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearmsg
        av.clearstatus
        exit
    end
end
'
for each iprec in ipftab
    ipshape = ipftab.returnvalue(ipshapef, iprec)
    ipline = ipshape.aspolyline
    ipcenter = ipshape.returncenter
    '
    outletlist = list.make
    for each hprec in hpftab
        hpshape = hpftab.returnvalue(hpshapef, hprec)
        if (ipline.distance(hpshape) < tol) then
            for each hlrec in hlftab
                hlshape = hlftab.returnvalue(hlshapef, hlrec)
                hlpoints = hlshape.asmultipoint.aslist
                if (hlpoints.count > 0) then
                    hltp = hlpoints.get(hlpoints.count - 1)
                    if (hltp.distance(hpshape) < tol) then
                        if (hlshape.iscontainedin(ipshape)) then
                            outletlist = outletlist.add(hprec.clone)
                            break
                        end
                    end
                end
            end
        end
    end
    if(outletlist.count = 0) then
        if (oblevel >= 2) then
            msgbox.error("Subbasin outlet not found." + nl + "Polygon Number "
+ iprec.asstring , "HECPREPRO")
        end
        errors = errors + 1
        continue
    end
    '
    if ((outletlist.count = 1) or (not igfound)) then
        outlet = outletlist.get(0)
    end
    '
    if ((outletlist.count > 1) and (igfound)) then
        zlow = 100000000
        for each outrec in outletlist
            hpshape = hpftab.returnvalue(hpshapef, outrec)
            zval = igtheme.returncellvalue(hpshape)
            if (zval <= zlow) then
                zlow = zval.clone
```

```
                outlet = outrec.clone
            end
        end
    end
    '
    outletshape = hpftab.returnvalue(hpshapef, outlet)
    hecdnid = hpftab.returnvalue(hphecidf, outlet)
    hecdnx = outletshape.getx
    hecdny = outletshape.gety
    '
    spdictrec = {hecid.clone, 5, 0, 1, ipcenter.getx.clone,
ipcenter.gety.clone, iprec.clone}
    spdict.add(hecid.clone, spdictrec.clone)
    dniddict.add(hecid.clone, {hecdnid.clone})
    '
    sldictrec = {0, 0, hecid.clone, hecdnid.clone, ipcenter.getx.clone,
ipcenter.gety.clone, hecdnx.clone, hecdny.clone}
    sldict.add(hecid.clone, sldictrec.clone)
    '
    hecid = hecid + 1
    '
end
'
'--------------------------------------
'--- create sym line shape file (9) ---
'--------------------------------------
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Creating sym line.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(97)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearstatus
        exit
    end
end
'
'--- set up sym line theme (9.1) ---
'
'create file
'
slfilename = av.getproject.makefilename("syml", "shp")
slftab = ftab.makenew(slfilename, polyline)
'
'get shape field
'
slshapef = slftab.findfield("shape")
'
'add fields to sym line theme
'
slfields1 = list.make
'
slhecidf = field.make("hecid", #FIELD_DECIMAL, 16, 4)
slhectypef = field.make("hectype", #FIELD_DECIMAL, 16, 4)
slhecupidf = field.make("hecupid", #FIELD_DECIMAL, 16, 4)
slhecdnidf = field.make("hecdnid", #FIELD_DECIMAL, 16, 4)
slhecupxf = field.make("hecupx", #FIELD_DECIMAL, 16, 4)
slhecupyf = field.make("hecupy", #FIELD_DECIMAL, 16, 4)
slhecdnxf = field.make("hecdnx", #FIELD_DECIMAL, 16, 4)
slhecdnyf = field.make("hecdny", #FIELD_DECIMAL, 16, 4)
'
slfields1 = slfields1.add(slhecidf)
slfields1 = slfields1.add(slhectypef)
slfields1 = slfields1.add(slhecupidf)
slfields1 = slfields1.add(slhecdnidf)
slfields1 = slfields1.add(slhecupxf)
slfields1 = slfields1.add(slhecupyf)
slfields1 = slfields1.add(slhecdnxf)
```

```
slfields1 = slfields1.add(slhecdnyf)
'
if (attrib) then
    slfields2 = list.make
    for each atfnrec in mapdict.get("reach").get(3)
        atf = ilftab.findfield(atfnrec.get(1))
        slfields2 = slfields2.add(atf)
    end
    slfields2.removeduplicates
    slfields1 = slfields1 + slfields2.deepclone
end
'
slftab.addfields(slfields1)
'
'make sym line theme editable
'
slftab.seteditable(true)
'
'--- user observation ---
'
if (oblevel >= 2) then
    sltheme = ftheme.make(slftab)
    theview.addtheme(sltheme)
    sltheme.setvisible(true)
    theview.draw(thedisplay)
end
'
'--- add lines (9.2) ---
'
for each slrec in sldict
    slfpointx = slrec.get(4)
    slfpointy = slrec.get(5)
    slfpoint = point.make(slfpointx, slfpointy)
    sltpointx = slrec.get(6)
    sltpointy = slrec.get(7)
    sltpoint = point.make(sltpointx, sltpointy)
    slline = polyline.make({{slfpoint, sltpoint}})
    '
    theoutrec = slftab.addrecord
    '
    slftab.setvalue(slshapef, theoutrec, slline)
    slftab.setvalue(slhecidf, theoutrec, slrec.get(0))
    slftab.setvalue(slhectypef, theoutrec, slrec.get(1))
    slftab.setvalue(slhecupidf, theoutrec, slrec.get(2))
    slftab.setvalue(slhecdnidf, theoutrec, slrec.get(3))
    slftab.setvalue(slhecupxf, theoutrec, slrec.get(4))
    slftab.setvalue(slhecupyf, theoutrec, slrec.get(5))
    slftab.setvalue(slhecdnxf, theoutrec, slrec.get(6))
    slftab.setvalue(slhecdnyf, theoutrec, slrec.get(7))
    '
    if (attrib) then
        reachlist = list.make
        for each hlrec in hlftab
            hlshape = hlftab.returnvalue(hlshapef, hlrec)
            hlpointlist = hlshape.asmultipoint.aslist
            if (hlpointlist.count < 2) then
                hllength = 0
            else
                fpoint = hlpointlist.get(0)
                tpoint = hlpointlist.get(hlpointlist.count - 1)
                hllength = (((tpoint.getx - fpoint.getx)^2) + ((tpoint.gety -
fpoint.gety)^2))^0.5
            end
            if (hllength > (tol / 2)) then
                hlhecid = hlftab.returnvalue(hlhecidf, hlrec)
                if (hlhecid = slrec.get(0)) then
                    reachlist = reachlist.add(hlrec.clone)
                end
            end
        end
        for each atfnrec in mapdict.get("reach").get(3)
```

```
            hlatf = hlftab.findfield(atfnrec.get(1))
            total = 0
            totnumb = 0
            totweight = 0
            for each hlrec in reachlist
                hlval = hlftab.returnvalue(hlatf, hlrec)
                if (atfnrec.get(2) = 1) then
                    total = total + hlval
                end
                if (atfnrec.get(2) = 2) then
                    total = total + hlval
                    totnumb = totnumb + 1
                end
                if (atfnrec.get(2) = 3) then
                    hlshape = hlftab.returnvalue(hlshapef, hlrec)
                    hlpointlist = hlshape.asmultipoint.aslist
                    i = 0
                    hllength = 0
                    while (true)
                        vxfp = hlpointlist.get(i)
                        vxtp = hlpointlist.get(i + 1)
                        vxl = (((vxtp.getx - vxfp.getx)^2) + ((vxtp.gety -
vxfp.gety)^2))^0.5
                        hllength = hllength + vxl
                        i = i + 1
                        if (i = (hlpointlist.count - 1)) then
                            break
                        end
                    end
                    total = total + (hlval * hllength)
                    totweight = totweight + hllength
                end
            end
            if (atfnrec.get(2) = 1) then
                slval = total
            end
            if (atfnrec.get(2) = 2) then
                slval = total / totnumb
            end
            if (atfnrec.get(2) = 3) then
                slval = total / totweight
            end
            slatf = slftab.findfield(atfnrec.get(1))
            slftab.setvalue(slatf, theoutrec, slval)
        end
    end
    '
    if (oblevel >= 3) then
        av.run("HECLEGEND", {sltheme})
        theview.draw(thedisplay)
    end
end
'
'--- user observation ---
'
if (oblevel >= 2) then
    av.run("HECLEGEND", {sltheme})
    theview.draw(thedisplay)
end
'
'---------------------------------------
'--- create sym point shape file (10) ---
'---------------------------------------
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Creating sym point.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(98)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
```

```
        msgbox.info(message, "HECPREPRO")
        av.clearstatus
        exit
    end
end
'
'--- set up sym point theme (10.1) ---
'
'create file
'
spfilename = av.getproject.makefilename("symp", "shp")
spftab = ftab.makenew(spfilename, point)
'
'get shape field
'
spshapef = spftab.findfield("shape")
'
'add fields to sym point theme
'
spfields1 = list.make
'
sphecidf = field.make("hecid", #FIELD_DECIMAL, 16, 4)
sphectypef = field.make("hectype", #FIELD_DECIMAL, 16, 4)
sphecupidsf = field.make("hecupids", #FIELD_DECIMAL, 16, 4)
sphecdnidsf = field.make("hecdnids", #FIELD_DECIMAL, 16, 4)
sphecxf = field.make("hecx", #FIELD_DECIMAL, 16, 4)
sphecyf = field.make("hecy", #FIELD_DECIMAL, 16, 4)
'
spfields1 = spfields1.add(sphecidf)
spfields1 = spfields1.add(sphectypef)
spfields1 = spfields1.add(sphecupidsf)
spfields1 = spfields1.add(sphecdnidsf)
spfields1 = spfields1.add(sphecxf)
spfields1 = spfields1.add(sphecyf)
'
if (attrib) then
    spfields2 = list.make
    for each maprec in mapdict.returnkeys
        if (maprec = "reach") then
            continue
        else
            for each atfnrec in mapdict.get(maprec).get(3)
                if (maprec = "subbasin") then
                    atf = ipftab.findfield(atfnrec.get(1))
                    spfields2 = spfields2.add(atf)
                else
                    if (infound) then
                        atf = inftab.findfield(atfnrec.get(1))
                        spfields2 = spfields2.add(atf)
                    end
                end
            end
        end
    end
    spfields2.removeduplicates
    spfields1 = spfields1 + spfields2.deepclone
end
'
spftab.addfields(spfields1)
'
'make sym point theme editable
'
spftab.seteditable(true)
'
'--- user observation ---
'
if (oblevel >= 2) then
    sptheme = ftheme.make(spftab)
    theview.addtheme(sptheme)
    sptheme.setvisible(true)
    theview.draw(thedisplay)
```

```
end
'
'--- add points (10.2) ---
'
for each sprec in spdict
    sppointx = sprec.get(4)
    sppointy = sprec.get(5)
    sppoint = point.make(sppointx, sppointy)
    theoutrec = spftab.addrecord
    spftab.setvalue(spshapef, theoutrec, sppoint)
    spftab.setvalue(sphecidf, theoutrec, sprec.get(0))
    spftab.setvalue(sphectypef, theoutrec, sprec.get(1))
    spftab.setvalue(sphecupidsf, theoutrec, sprec.get(2))
    spftab.setvalue(sphecdnidsf, theoutrec, sprec.get(3))
    spftab.setvalue(sphecxf, theoutrec, sprec.get(4))
    spftab.setvalue(sphecyf, theoutrec, sprec.get(5))
    if (attrib) then
        if (sprec.get(1) = 5) then
            for each atfnrec in mapdict.get("subbasin").get(3)
                ipatf = ipftab.findfield(atfnrec.get(1).clone)
                ipval = ipftab.returnvalue(ipatf, sprec.get(6))
                spatf = spftab.findfield(atfnrec.get(1).clone)
                spftab.setvalue(spatf, theoutrec, ipval)
            end
        else
            if (infound) then
                for each herec in hedict.returnkeys
                    if (sprec.get(1) = herec) then
                        hename = hedict.get(herec)
                        for each atfnrec in mapdict.get(hename).get(3)
                            hpatf = hpftab.findfield(atfnrec.get(1))
                            hpval = hpftab.returnvalue(hpatf, sprec.get(6))
                            spatf = spftab.findfield(atfnrec.get(1))
                            spftab.setvalue(spatf, theoutrec, hpval)
                        end
                    end
                end
            end
        end
    end
    '
    if (oblevel >= 2) then
        av.run("HECLEGEND", {sptheme})
        theview.draw(thedisplay)
    end
end
'
'--- user observation ---
'
if (oblevel >= 2) then
    av.run("HECLEGEND", {sptheme})
    theview.draw(thedisplay)
end
'
'------------------------
'--- link tables (11) ---
'------------------------
'
slftab.link(slhecidf, hlftab, hlhecidf)
spftab.link(sphecidf, hpftab, hphecidf)
'
'--------------------------------------
'--- create HEC-HMS basin file (12) ---
'--------------------------------------
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Creating HMS basin file.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(99)
    if (not keepgoing) then
```

```
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearstatus
        exit
    end
end
'
'--- set up file (12.1) ---
'
if (hmsmode = "d") then
    hmsfilename = av.getproject.makefilename("basin", "txt")
else
    hmsfilename = hmsmode.asfilename
end
hmsfile = linefile.make(hmsfilename, #FILE_PERM_WRITE)
'
'--- write header (12.2) ---
'
outstring = "Basin: HECPREPRO generated HMS basin file"
hmsfile.writeelt(outstring)
outstring = "     Description: HECPREPRO Version 4.0.av"
hmsfile.writeelt(outstring)
thisdate = date.now.setformat("dd MMMM yyyy")
outstring = "     Last Modified Date: " + thisdate.asstring
hmsfile.writeelt(outstring)
thistime = date.now.setformat("hhh:m")
outstring = "     Last Modified Time: " + thistime.asstring
hmsfile.writeelt(outstring)
outstring = "     Unit System: Unknown"
hmsfile.writeelt(outstring)
outstring = "End:"
hmsfile.writeelt(outstring)
outstring = ""
hmsfile.writeelt(outstring)
'
'--- write element data (12.3) ---
'
'subbasin elements
'
for each sprec in spftab
    sphectype = spftab.returnvalue(sphectypef, sprec)
    if (sphectype = 5) then
        sphecid = spftab.returnvalue(sphecidf, sprec)
        outstring = "Subbasin: " + sphecid.asstring
        hmsfile.writeelt(outstring)
        '
        sphecx = spftab.returnvalue(sphecxf, sprec)
        outstring = "     Canvas X: " + sphecx.asstring
        hmsfile.writeelt(outstring)
        '
        sphecy = spftab.returnvalue(sphecyf, sprec)
        outstring = "     Canvas Y: " + sphecy.asstring
        hmsfile.writeelt(outstring)
        '
        sphecdnidlist = dniddict.get(sphecid)
        outstring = "     Downstream: "
        for each dnid in sphecdnidlist
            outstring = outstring ++ dnid.asstring
        end
        hmsfile.writeelt(outstring)
        '
        if (attrib) then
            for each atfrec in mapdict.get("subbasin").get(3)
                spf = spftab.findfield(atfrec.get(1))
                spval = spftab.returnvalue(spf, sprec)
                outstring = "     " + atfrec.get(0) + ": " + spval.asstring
                hmsfile.writeelt(outstring)
            end
        end
        '
```

```
        outstring = "End:"
        hmsfile.writeelt(outstring)
        '
        outstring = ""
        hmsfile.writeelt(outstring)
    end
end
'
'source elements
'
for each sprec in spftab
    sphectype = spftab.returnvalue(sphectypef, sprec)
    if (sphectype = 7) then
        sphecid = spftab.returnvalue(sphecidf, sprec)
        outstring = "Source: " + sphecid.asstring
        hmsfile.writeelt(outstring)
        '
        sphecx = spftab.returnvalue(sphecxf, sprec)
        outstring = "     Canvas X: " + sphecx.asstring
        hmsfile.writeelt(outstring)
        '
        sphecy = spftab.returnvalue(sphecyf, sprec)
        outstring = "     Canvas Y: " + sphecy.asstring
        hmsfile.writeelt(outstring)
        '
        sphecdnidlist = dniddict.get(sphecid)
        outstring = "     Downstream: "
        for each dnid in sphecdnidlist
            outstring = outstring ++ dnid.asstring
        end
        hmsfile.writeelt(outstring)
        '
        if (attrib) then
            for each atfrec in mapdict.get("source").get(3)
                spf = spftab.findfield(atfrec.get(1))
                spval = spftab.returnvalue(spf, sprec)
                outstring = "     " + atfrec.get(0) + ": " + spval.asstring
                hmsfile.writeelt(outstring)
            end
        end
        '
        outstring = "End:"
        hmsfile.writeelt(outstring)
        '
        outstring = ""
        hmsfile.writeelt(outstring)
    end
end
'
'reach elements
'
for each slrec in slftab
    slhectype = slftab.returnvalue(slhectypef, slrec)
    if (slhectype = 4) then
        slhecid = slftab.returnvalue(slhecidf, slrec)
        outstring = "Reach: " + slhecid.asstring
        hmsfile.writeelt(outstring)
        '
        slhecdnx = sldict.get(slhecid).get(6)
        outstring = "     Canvas X: " + slhecdnx.asstring
        hmsfile.writeelt(outstring)
        '
        slhecdny = sldict.get(slhecid).get(7)
        outstring = "     Canvas Y: " + slhecdny.asstring
        hmsfile.writeelt(outstring)
        '
        slhecupx = sldict.get(slhecid).get(4)
        outstring = "     From Canvas X: " + slhecupx.asstring
        hmsfile.writeelt(outstring)
        '
        slhecupy = sldict.get(slhecid).get(5)
```

```
        outstring = "     From Canvas Y: " + slhecupy.asstring
        hmsfile.writeelt(outstring)
        '
        slhecdnid = slftab.returnvalue(slhecdnidf, slrec)
        outstring = "     Downstream: " + slhecdnid.asstring
        hmsfile.writeelt(outstring)
        '
        if (attrib) then
            reachlist = list.make
            for each hlrec in hlftab
                hlshape = hlftab.returnvalue(hlshapef, hlrec)
                hlpointlist = hlshape.asmultipoint.aslist
                if (hlpointlist.count < 2) then
                    hllength = 0
                else
                    fpoint = hlpointlist.get(0)
                    tpoint = hlpointlist.get(hlpointlist.count - 1)
                    hllength = (((tpoint.getx - fpoint.getx)^2) +
((tpoint.gety - fpoint.gety)^2))^0.5
                end
                if (hllength > (tol / 2)) then
                    hlhecid = hlftab.returnvalue(hlhecidf, hlrec)
                    if (hlhecid = slhecid) then
                        reachlist = reachlist.add(hlrec.clone)
                    end
                end
            end
            for each atfnrec in mapdict.get("reach").get(3)
                hlf = hlftab.findfield(atfnrec.get(1))
                total = 0
                totnumb = 0
                totweight = 0
                for each hlrec in reachlist
                    hlval = hlftab.returnvalue(hlf, hlrec)
                    if (atfnrec.get(2) = 1) then
                        total = total + hlval
                    end
                    if (atfnrec.get(2) = 2) then
                        total = total + hlval
                        totnumb = totnumb + 1
                    end
                    if (atfnrec.get(2) = 3) then
                        hlshape = hlftab.returnvalue(hlshapef, hlrec)
                        hlpointlist = hlshape.asmultipoint.aslist
                        i = 0
                        hllength = 0
                        while (true)
                            vxfp = hlpointlist.get(i)
                            vxtp = hlpointlist.get(i + 1)
                            vxl = (((vxtp.getx - vxfp.getx)^2) + ((vxtp.gety -
vxfp.gety)^2))^0.5
                            hllength = hllength + vxl
                            i = i + 1
                            if (i = (hlpointlist.count - 1)) then
                                break
                            end
                        end
                        total = total + (hlval * hllength)
                        totweight = totweight + hllength
                    end
                end
                if (atfnrec.get(2) = 1) then
                    slval = total
                end
                if (atfnrec.get(2) = 2) then
                    slval = total / totnumb
                end
                if (atfnrec.get(2) = 3) then
                    slval = total / totweight
                end
                outstring = "     " + atfnrec.get(0) + ": " + slval.asstring
```

```
                hmsfile.writeelt(outstring)
            end
        end
        '
        outstring = "End:"
        hmsfile.writeelt(outstring)
        '
        outstring = ""
        hmsfile.writeelt(outstring)
    end
end
'
'junction elements
'
for each sprec in spftab
    sphectype = spftab.returnvalue(sphectypef, sprec)
    if ((sphectype = 3) or (sphectype = 1)) then
        sphecid = spftab.returnvalue(sphecidf, sprec)
        outstring = "Junction: " + sphecid.asstring
        hmsfile.writeelt(outstring)
        '
        sphecx = spftab.returnvalue(sphecxf, sprec)
        outstring = "     Canvas X: " + sphecx.asstring
        hmsfile.writeelt(outstring)
        '
        sphecy = spftab.returnvalue(sphecyf, sprec)
        outstring = "     Canvas Y: " + sphecy.asstring
        hmsfile.writeelt(outstring)
        '
        sphecdnidlist = dniddict.get(sphecid)
        outstring = "     Downstream: "
        for each dnid in sphecdnidlist
            outstring = outstring ++ dnid.asstring
        end
        hmsfile.writeelt(outstring)
        '
        if (attrib) then
            for each atfrec in mapdict.get("junction").get(3)
                spf = spftab.findfield(atfrec.get(1))
                spval = spftab.returnvalue(spf, sprec)
                outstring = "     " + atfrec.get(0) + ": " + spval.asstring
                hmsfile.writeelt(outstring)
            end
        end
        '
        outstring = "End:"
        hmsfile.writeelt(outstring)
        '
        outstring = ""
        hmsfile.writeelt(outstring)
    end
end
'
'reservoir elements
'
for each sprec in spftab
    sphectype = spftab.returnvalue(sphectypef, sprec)
    if (sphectype = 10) then
        sphecid = spftab.returnvalue(sphecidf, sprec)
        outstring = "Reservoir: " + sphecid.asstring
        hmsfile.writeelt(outstring)
        '
        sphecx = spftab.returnvalue(sphecxf, sprec)
        outstring = "     Canvas X: " + sphecx.asstring
        hmsfile.writeelt(outstring)
        '
        sphecy = spftab.returnvalue(sphecyf, sprec)
        outstring = "     Canvas Y: " + sphecy.asstring
        hmsfile.writeelt(outstring)
        '
        sphecdnidlist = dniddict.get(sphecid)
```

```
        outstring = "     Downstream: "
        for each dnid in sphecdnidlist
            outstring = outstring ++ dnid.asstring
        end
        hmsfile.writeelt(outstring)
        '
        if (attrib) then
            for each atfrec in mapdict.get("reservoir").get(3)
                spf = spftab.findfield(atfrec.get(1))
                spval = spftab.returnvalue(spf, sprec)
                outstring = "     " + atfrec.get(0) + ": " + spval.asstring
                hmsfile.writeelt(outstring)
            end
        end
        '
        outstring = "End:"
        hmsfile.writeelt(outstring)
        '
        outstring = ""
        hmsfile.writeelt(outstring)
    end
end
'
'diversion elements
'
for each sprec in spftab
    sphectype = spftab.returnvalue(sphectypef, sprec)
    if (sphectype = 8) then
        sphecid = spftab.returnvalue(sphecidf, sprec)
        outstring = "Diversion: " + sphecid.asstring
        hmsfile.writeelt(outstring)
        '
        sphecx = spftab.returnvalue(sphecxf, sprec)
        outstring = "     Canvas X: " + sphecx.asstring
        hmsfile.writeelt(outstring)
        '
        sphecy = spftab.returnvalue(sphecyf, sprec)
        outstring = "     Canvas Y: " + sphecy.asstring
        hmsfile.writeelt(outstring)
        '
        sphecdnidlist = dniddict.get(sphecid)
        outstring = "     Downstream: "
        for each dnid in sphecdnidlist
            outstring = outstring ++ dnid.asstring
        end
        hmsfile.writeelt(outstring)
        '
        if (attrib) then
            for each atfrec in mapdict.get("diversion").get(3)
                spf = spftab.findfield(atfrec.get(1))
                spval = spftab.returnvalue(spf, sprec)
                outstring = "     " + atfrec.get(0) + ": " + spval.asstring
                hmsfile.writeelt(outstring)
            end
        end
        '
        outstring = "End:"
        hmsfile.writeelt(outstring)
        '
        outstring = ""
        hmsfile.writeelt(outstring)
    end
end
'
'sink elements
'
for each sprec in spftab
    sphectype = spftab.returnvalue(sphectypef, sprec)
    if (sphectype = 6) then
        sphecid = spftab.returnvalue(sphecidf, sprec)
        outstring = "Sink: " + sphecid.asstring
```

```
        hmsfile.writeelt(outstring)
        '
        sphecx = spftab.returnvalue(sphecxf, sprec)
        outstring = "     Canvas X: " + sphecx.asstring
        hmsfile.writeelt(outstring)
        '
        sphecy = spftab.returnvalue(sphecyf, sprec)
        outstring = "     Canvas Y: " + sphecy.asstring
        hmsfile.writeelt(outstring)
        '
        if (attrib) then
            for each atfrec in mapdict.get("sink").get(3)
                spf = spftab.findfield(atfrec.get(1))
                spval = spftab.returnvalue(spf, sprec)
                outstring = "     " + atfrec.get(0) + ": " + spval.asstring
                hmsfile.writeelt(outstring)
            end
        end

        '
        outstring = "End:"
        hmsfile.writeelt(outstring)
        '
        outstring = ""
        hmsfile.writeelt(outstring)
    end
end
'
'--- close file (12.4) ---
'
hmsfile.close
'
'---------------------
'--- close up (13) ---
'---------------------
'
'--- user observation ---
'
if (oblevel >= 1) then
    mainmsg = "HECPREPRO: Closing up.  "
    av.showmsg(mainmsg)
    keepgoing = av.setstatus(100)
    if (not keepgoing) then
        message = "Premature end." + nl + nl + "Errors detected = " +
errors.asstring
        msgbox.info(message, "HECPREPRO")
        av.clearstatus
        exit
    end
end
'
'--- make themes non-editable ---
'
hlftab.seteditable(false)
hpftab.seteditable(false)
slftab.seteditable(false)
spftab.seteditable(false)
'
'--- final message to user ---
'
if (oblevel >= 1) then
    av.showmsg("HECPREPRO: Normal end.")
    av.setstatus(100)
    message = "Normal end."
    message = message + nl + "Errors detected = " + errors.asstring
    message = message + nl + "HMS basin file saved to " + hmsfilename.asstring
    msgbox.info(message, "HECPREPRO")
    av.clearmsg
    av.clearstatus
end
'
```

```
'-----------
'--- end ---
'-----------
'
```

B.2. Legend Utility (heclegend.ave).

```
'
'-----------------
'-----------------
'--- HECLEGEND ---
'-----------------
'-----------------
'
'----------------------------
'--- creation information ---
'----------------------------
'
'Name: heclegend.ave
'Version: 4.0.av.na
'Date: 02/22/97
'Author: Ferdi Hellweger
'        Center for Research in Water Resources
'        The University of Texas at Austin
'        ferdi@crwr.utexas.edu
'        www.ce.utexas.edu/stu/ferdi/
'
'---------------------------
'--- purpose/description ---
'---------------------------
'
'This script creates/updates the legend for HECPREPRO themes.
'
'-----------------
'--- get theme ---
'-----------------
'
if ((self = nil) or (self.getclass.getclassname = "button")) then
    theview = av.getactivedoc
    theactivethemes = theview.getactivethemes
    if (theactivethemes.count = 0) then
        msgbox.error("No active themes found", "HECPREPRO")
        exit
    end
    if (theactivethemes.count > 1) then
        msgbox.error("Too many active themes found", "HECPREPRO")
        exit
    end
    thetheme = theactivethemes.get(0)
else
    thetheme = self.get(0)
end
'
'------------------
'--- get legend ---
'------------------
'
thelegend = thetheme.getlegend
'
'-----------------------
'--- set legend type ---
'-----------------------
'
thelegend.unique(thetheme, "hectype")
'
'---------------------------------
'--- get legend classifications ---
```

```
'---------------------------------
'
lcs = thelegend.getclassifications
'
'---------------------------------
'--- set legend classifications ---
'---------------------------------
'
darkgreen = color.make
darkgreen.setrgblist({50, 155, 50})
'
for each lc in lcs
    lcv = lc.returnmaximum
    if (lcv = 0) then
        lc.setlabel("none")
        thelegend.getsymbol({lcv},false).setcolor(color.getblack)
    end
    if (lcv = 1) then
        lc.setlabel("outlet")
        thelegend.getsymbol({lcv},false).setcolor(color.getcyan)
    end
    if (lcv = 2) then
        lc.setlabel("error")
        thelegend.getsymbol({lcv},false).setcolor(color.getblack)
    end
    if (lcv = 3) then
        lc.setlabel("junction")
        thelegend.getsymbol({lcv},false).setcolor(darkgreen)
    end
    if (lcv = 4) then
        lc.setlabel("channel")
        thelegend.getsymbol({lcv},false).setcolor(color.getgreen)
    end
    if (lcv = 5) then
        lc.setlabel("subbasin")
        thelegend.getsymbol({lcv},false).setcolor(color.getgray)
    end
    if (lcv = 6) then
        lc.setlabel("sink")
        thelegend.getsymbol({lcv},false).setcolor(color.getred)
    end
    if (lcv = 7) then
        lc.setlabel("source")
        thelegend.getsymbol({lcv},false).setcolor(color.getgreen)
    end
    if (lcv = 8) then
        lc.setlabel("diversion")
        thelegend.getsymbol({lcv},false).setcolor(color.getmagenta)
    end
    if (lcv = 9) then
        lc.setlabel("error")
        thelegend.getsymbol({lcv},false).setcolor(color.getblack)
    end
    if (lcv >= 10) then
        lc.setlabel("lake")
        thelegend.getsymbol({lcv},false).setcolor(color.getblue)
    end
end
'
'---------------------
'--- update legend ---
'---------------------
'
thetheme.updatelegend
'
'-----------
'--- end ---
'-----------
'
```

APPENDIX C. SAMPLE OUTPUT.

C.1. HEC-HMS Basin File.

```
Basin: HECPREPRO generated HMS basin file
     Description: HECPREPRO generated HMS basin file
     Last Modified Date: 18 June 1996
     Last Modified Time: 14:54
     Unit System: Unknown
End:

Subbasin:  23
     Canvas X:  115488.609
     Canvas Y:  1455012.500
     Downstream:  7
     Area:  845179904.
End:

Subbasin:  22
     Canvas X:  149706.500
     Canvas Y:  1453787.875
     Downstream:  2
     Area:  1556919808.
End:

Subbasin:  24
     Canvas X:  126467.914
     Canvas Y:  1431180.625
     Downstream:  6
     Area:  824919808.
End:

Subbasin:  25
     Canvas X:  100018.953
     Canvas Y:  1415211.500
     Downstream:  11
     Area:  936280000.
End:

Source:  1
     Canvas X:  169282.81250
     Canvas Y:  1462399.39734
     Downstream:  12
End:

Source:  3
     Canvas X:  99182.82031
     Canvas Y:  1453975.99966
     Downstream:  14
End:

Reach:  12
     Downstream:  2
End:

Reach:  13
     Downstream:  5
End:

Reach:  14
     Downstream:  5
End:

Reach:  16
     Downstream:  7
End:

Reach:  15
     Downstream:  4
End:

Reach:  17
```

```
     Downstream:  8
End:

Reach:  18
     Downstream:  8
End:

Reach:  19
     Downstream:  9
End:

Reach:  20
     Downstream:  10
End:

Reach:  21
     Downstream:  11
End:

Junction:  2
     Canvas X:  127482.82031
     Canvas Y:  1454141.12500
     Downstream:  13
End:

Junction:  6
     Canvas X:  103982.82031
     Canvas Y:  1430399.77344
     Downstream:  17
End:

Junction:  7
     Canvas X:  96982.82031
     Canvas Y:  1430299.77344
     Downstream:  18
End:

Junction:  8
     Canvas X:  97424.17188
     Canvas Y:  1420841.12500
     Downstream:  19
End:

Reservoir:  5
     Canvas X:  99024.17188
     Canvas Y:  1438141.12500
     Downstream:  16 15
End:

Diversion:  9
     Canvas X:  93124.17188
     Canvas Y:  1406660.84829
     Downstream:  20 21
End:

Sink:  4
     Canvas X:  93081.76151
     Canvas Y:  1439792.12091
End:

Sink:  10
     Canvas X:  98778.64854
     Canvas Y:  1399398.62500
End:

Sink:  11
     Canvas X:  85182.82031
     Canvas Y:  1393441.12500
End:
```

C.2. General Text File.

```
HECPREPRO: OUTPUT FILE
Created 06/18/96.14:54:09.Tue
-----------------
--- SUBBASINS ---
-----------------
---
HECID:
23
X-COORD:
115488.609
Y-COORD:
1455012.500
CENZ:
274.00
B-COORD:
bc3
F-COORD:
fc3
FUPZ:
304.00
FLENGTH:
59209.62
FLENGTH2:
1760.66
AREA:
845179904.
MEANS:
0.038411
MEDIANS:
0.021000
HECDNID:
7
---
HECID:
22
X-COORD:
149706.500
Y-COORD:
1453787.875
CENZ:
365.00
B-COORD:
bc2
F-COORD:
fc2
FUPZ:
487.00
FLENGTH:
64474.01
FLENGTH2:
28578.23
AREA:
1556919808.
MEANS:
0.031724
MEDIANS:
0.016000
HECDNID:
2
---
HECID:
24
X-COORD:
126467.914
Y-COORD:
1431180.625
```

CENZ:
324.00
B-COORD:
bc4
F-COORD:
fc4
FUPZ:
334.00
FLENGTH:
41337.37
FLENGTH2:
27262.13
AREA:
824919808.
MEANS:
0.040621
MEDIANS:
0.026000
HECDNID:
6

HECID:
25
X-COORD:
100018.953
Y-COORD:
1415211.500
CENZ:
243.00
B-COORD:
bc5
F-COORD:
fc5
FUPZ:
242.00
FLENGTH:
53012.73
FLENGTH2:
27123.33
AREA:
936280000.
MEANS:
0.041750
MEDIANS:
0.024000
HECDNID:
11

--- SOURCES ---

HECID:
1
X-COORD:
169282.81250
Y-COORD:
1462399.39734
NODEZ:
487.00
HECDNID:
12

HECID:
3
X-COORD:
99182.82031
Y-COORD:
1453975.99966
NODEZ:
353.00
HECDNID:

14

--- REACHES ---

HECID:
12
HECLENGTH:
64474.01
HECSLOPE:
0.003133
HECUPID:
1
HECDNID:
2

HECID:
13
HECLENGTH:
29577.74
HECSLOPE:
0.001420
HECUPID:
2
HECDNID:
5

HECID:
14
HECLENGTH:
7227.87
HECSLOPE:
0.015219
HECUPID:
3
HECDNID:
5

HECID:
16
HECLENGTH:
13012.39
HECSLOPE:
0.002305
HECUPID:
5
HECDNID:
7

HECID:
15
HECLENGTH:
4081.98
HECSLOPE:
-.017149
HECUPID:
0
HECDNID:
4

HECID:
17
HECLENGTH:
12568.22
HECSLOPE:
0.002307
HECUPID:
6
HECDNID:
8

HECID:
18
HECLENGTH:
10825.58
HECSLOPE:
0.000000
HECUPID:
7
HECDNID:
8

HECID:
19
HECLENGTH:
20989.82
HECSLOPE:
0.001477
HECUPID:
8
HECDNID:
9

HECID:
20
HECLENGTH:
9969.68
HECSLOPE:
-.013441
HECUPID:
9
HECDNID:
10

HECID:
21
HECLENGTH:
19454.68
HECSLOPE:
0.001542
HECUPID:
9
HECDNID:
11

--- JUNCTIONS ---

HECID:
2
X-COORD:
127482.82031
Y-COORD:
1454141.12500
NODEZ:
285.00
HECUPID(S):
12
22
HECDNID:
13

HECID:
6
X-COORD:
103982.82031
Y-COORD:
1430399.77344
NODEZ:
242.00
HECUPID(S):
24

HECDNID:
17

HECID:
7
X-COORD:
96982.82031
Y-COORD:
1430299.77344
NODEZ:
213.00
HECUPID(S):
16
23
HECDNID:
18

HECID:
8
X-COORD:
97424.17188
Y-COORD:
1420841.12500
NODEZ:
213.00
HECUPID(S):
17
18
HECDNID:
19

--- RESERVOIRS ---

HECID:
5
X-COORD:
99024.17188
Y-COORD:
1438141.12500
NODEZ:
243.00
HECUPID(S):
13
14
HECDNID(S):
16
15

--- DIVERSIONS ---

HECID:
9
X-COORD:
93124.17188
Y-COORD:
1406660.84829
NODEZ:
182.00
HECUPID:
19
HECDNIDS:
20
21

--- SINKS ---

HECID:
4

X-COORD:
93081.76151
Y-COORD:
1439792.12091
NODEZ:
313.00
HECUPID(S):
15

HECID:
10
X-COORD:
98778.64854
Y-COORD:
1399398.62500
NODEZ:
316.00
HECUPID(S):
20

HECID:
11
X-COORD:
85182.82031
Y-COORD:
1393441.12500
NODEZ:
152.00
HECUPID(S):
21
25

--- END ---
----

8. REFERENCES.

Bhaskar, N. R., James, W. P., Devulapalli, R. S., Hydrologic Parameter Estimation using Geographic Information System, Journal of Water Resources Planning and Management, vol. 118, no. 5, p. 492-512, 1992.

Charley, W., Pabst, A., and Peters, J., The Hydrologic Modeling System (HEC-HMS): Design and Development Issues, Technical Paper No. 149, Hydrologic Engineering Center, US Army Corps of Engineers, June 1995.

DeVantier, B. A., and Feldman, A. D., Review of GIS Applications in Hydrologic Modeling, Journal of Water Resources Planning and Management, vol. 119, no. 2, p. 246-261, 1993.

Engineering Computer Graphics Laboratory (ECGL), Watershed Modeling System, Brigham Young University, 1997.

Environmental Systems Research Institute, Inc. (ESRI), ARC/INFO Version 7.0.4, Redlands, CA, 1996.

Environmental Systems Research Institute, Inc. (ESRI), ArcView GIS Version 3.0, Redlands, CA, 1996.

Environmental Systems Research Institute, Inc. (ESRI), What is a GIS, http://www.esri.com/base/gis/abtgis/what_gis.html, 5/21/97, 1997.

Fisher, G. T., Geographic Information System/Watershed Model Interface, Proceedings of the 1989 National Conference on Hydraulic Engineering, ASCE, New York, 1989.

Hydrologic Engineering Center (HEC), HEC-HMS: Hydrologic Modeling System, User's Manual (DRAFT), Hydrologic Engineering Center, US Army Corps of Engineers, Davis, California, September 1996.

Innovative System Developers, Inc., Geo-STORM Hydrology, Version 1.5 Beta, Columbia, MD, 1995.

Maidment, D. R., Developing a Watershed Data Structure, prepared for the Hydrologic Engineering Center, US Army Corps of Engineers, Davis, Calif., under Contract DACW05-92-P-1864, April 9, 1993.

Maidment, D. R., GIS and Hydrologic Modeling - an Assessment of Progress, The Third International Conference on GIS and Environmental Modeling, Santa Fe, New Mexico, 1996.

Mizgalewicz, P. J., and Maidment, D. R., The SAST Flood Water Balance (1993), Center for Research in Water Resources, University of Texas at Austin, September 9, 1996.

Moore, I. D., Hydrologic Modeling and GIS, in Environmental Modeling with GIS, editors Goodchild, M. F., Parks, B. O., and Steyaert, L. T., Oxford University, New York, 1993.

Olivera, F., McKinney, D. C., Maidment, D. R., Ye, Z., and Reed, S., Mean-Annual Water Balance of the Niger River, West Africa: Predicting the Water Balance of Surface and Ground Water Resources Over Large Areas, UNESCO Symposium on Runoff Computations for Water Projects, St. Petersberg, Russia, Oct. 30 - Nov. 3, 1995.

Olivera, F., Maidment, D. R., and Charbeneau, R. J., Spatially Distributed Modeling of Storm Runoff and Non-Point Source Pollution Using Geographic Information Systems, Center for Research in Water Resources, The University of Texas at Austin, 1996.

Peters, J., The HEC Hydrologic Modeling System, Technical Paper No. 150, Hydrologic Engineering Center, US Army Corps of Engineers, November 1995.

Reed, S., Maidment, D., Patoux, J., Spatial Water Balance of Texas, Center for Research in Water Resources, The University of Texas at Austin, 2/23/97.

Saghafian, B., Implementation of a Distributed Hydrologic Model within GRASS, GIS and Environmental Modeling: Progress and Research Issues, Goodchild, M. F., Steyaert, L. T., Parks, B. O., Johnston, C., Maidment, D., Crane, M., Glendinning, S., Editors, GIS World Books, Fort Collins, CO, 1996.

Shea, C., Grayman, W., Darden, D., Males, R. M., Sushinsky, P., Integrated GIS and Hydrologic Modeling for Countywide Drainage Study, Journal of Water Resources Planning and Management, vol. 119, no. 2, p. 112-128, 1993.

Smith, P. N., and Maidment, D. R., Hydrologic Data Development System, CRWR Online Report 95-1, Center for Research in Water Resource, The University of Texas at Austin, Austin, TX, 1995.

Stuebe, M. M., Johnston, D. M., Runoff Volume Estimation Using GIS Techniques, Water Resources Bulletin, vol. 26, no. 4, p. 611-620, 1990.

Warwick, J. J., Haness, S. J., Efficacy of ARC/INFO GIS Application to Hydrologic Modeling, Journal of Water Resources Planning and Management, vol. 120, no. 3, p. 366-381, 1994.

VITA

Ferdinand Leberecht Hellweger was born in Tuebingen, Germany on September 25, 1971, the son of Barbara Hellweger and Sebastian Raimund Hellweger. In September 1989 he entered the United States of America as a high school exchange student. After completing his senior year at King Philip Regional High School, Wrentham, Massachusetts, in 1990, he entered Northeastern University in Boston, Massachusetts. During his time at Northeastern University he worked for the City of Cambridge, Department of Public Works, Engineering and Sewer Division, Cambridge, Massachusetts and Cochrane Associates, Incorporated, Boston, Massachusetts. He received the degree of Bachelor of Science in 1995. In September, 1995, he entered The Graduate School at The University of Texas at Austin.

Permanent Address: 2103 Nueces

Austin, Texas 78705

This thesis was typed by the author.